DAHLGREN DIVISION
NAVAL SURFACE WARFARE CENTER
Dahlgren, Virginia 22448-5100

NSWCDD/MP-98/118

IMPLEMENTATION OF SUCCESSFUL PRACTICES USING AN ITERATIVE DEVELOPMENT METHODOLOGY FOR AN AEGIS CONFIGURATION MANAGEMENT SOFTWARE APPLICATION

BY SHARON N. COLSTON

COMBAT SYSTEMS DEPARTMENT

SEPTEMBER 1998

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1998	Final

4. TITLE AND SUBTITLE

Implementation of Successful Practices Using an Iterative Development Methodology for an AEGIS Configuration Management Software Application

5. FUNDING NUMBERS

6. AUTHOR(s)

Sharon N. Colston

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Commander
Naval Surface Warfare Center
Dahlgren Division (Code N27)
17320 Dahlgren Road
Dahlgren, VA 22448-5100

8. PERFORMING ORGANIZATION REPORT NUMBER

NSWCDD/MP-98/118

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This paper documents a two-and-a-half year software development project of the Combat Systems Configuration Management Branch of the Combat Systems Department at Naval Surface Warfare Center, Dahlgren Division (NSWCDD), at Dahlgren, Virginia. The author simultaneously managed two AEGIS Configuration Control Engineering Status System (ACCESS) software development projects and pursued a master's degree in Software Engineering Technical Management through the National Technological University sponsored by NSWCDD.

The paper describes lessons learned and practical experience gained by the author in managing a software development project to successful completion. The narrative includes methods that worked and ones that did not work. Critical success factors that the author considers most important to satisfactory software development are: (1) structured and documented iterative lifecycle development methodology; (2) implementation of software engineering repeatable processes; (3) highly-motivated, technically experienced dedicated team; (4) continuous and substantial customer involvement; (5) self-directed, empowered management style; and (6) experienced and committed project leader.

14. SUBJECT TERMS

Lifecycle Development Methodology, Capability Maturity Model, quality assurance, data transition, team motivation, customer involvement

15. NUMBER OF PAGES
156

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORTS	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

FOREWORD

This paper provides a compilation of practical knowledge and techniques that can be employed to facilitate successful and timely delivery of quality software. The paper is based on the results of a specific N Department development effort; however, the information is general in nature and can be used in other software development efforts. The goal is to provide knowledge gained from actual experiences, which might help another project leader avoid months, perhaps years, of trial and error and disappointing results. Also presented are specific techniques for use in delivering quality software that meets customer expectations.

Those interested in the following areas should find this document a useful reference tool:

- New Approaches to Lifecycle Development Methodologies
- Quality Assurance and Process Improvement
- Team Motivation and Development
- Project Management Techniques

Approved by:

L. M. WILLIAMS, III, Head
Combat Systems Department

CONTENTS

CONTENTS (Continued)

CONTENTS (Continued)

ILLUSTRATIONS

TABLES

GLOSSARY

ACCESS	AEGIS Configuration Control Engineering Status System
ACS	AEGIS Combat System
ANSI	American National Standards Institute
AWS	AEGIS Weapons System
BL	Baseline
CASE	Computer Aided Software Engineering
CC	Change Control
CM	Configuration Management
CMM	Capability Maturity Model
COTS	Commercial Off The Shelf
CP	Computer Program
CPCR	Computer Program Change Request
CPM	Critical Path Method
CRB	Change Review Board
DOD	Department of Defense
FLCR	Film Label Change Request
GUI	Graphical User Interface
ICR	Interface Change Request
IE	Information Engineering
JAD	Joint Application Development
KPA	Key Process Area
LOC	Lines of Code
NSWCDD	Naval Surface Warfare Center, Dahlgren Division
PERT	Program Evaluation and Review Technique
PL/SQL	Procedural Language/Structured Query Language
PVCS	Production Version Control System
RAD	Rapid Application Development
SC	Specification Change
SEI	Software Engineering Institute
SOW	Statement of Work
SPOC	Security Point of Contact
SQA	Software Quality Assurance
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UAT	User Acceptance Testing

SECTION 1

INTRODUCTION

From 1993 through 1997, the author pursued a master's degree in Software Engineering Technical Management through the National Technological University sponsored by the Naval Surface Warfare Center, Dahlgren Division (NSWCDD). During that time, this author managed two AEGIS Configuration Control Engineering Status System (ACCESS) software development projects and had the opportunity to apply much of what was learned in pursuit of the degree in managing these projects. This paper documents the synthesis of formal education and practical experience on the most recent software development project, which spanned a period of two and one-half years.

The narrative addresses four areas that impacted the project:

- Life Cycle Development Methodology
- Quality Assurance and Process Improvement
- Project Management
- Team Motivation

The discussion of each of these areas includes background information, implementation efforts, and results supported by material from leading experts in the area of technical management.

The purpose of this paper is to present a compilation of practical knowledge and techniques that can be employed to facilitate successful and timely delivery of quality software. This paper concentrates on what worked in this particular project; however, the paper also documents what did not work. It is the goal of the author to provide knowledge that might help the reader avoid months, perhaps years, of trial and error and disappointing results. Although the information supplied relates to a specific project with specific constraints, much of the information is general in nature and should be of interest to other project managers leading software development efforts.

SECTION 2

BACKGROUND

The Combat Systems Configuration Management Branch, N27, within the Combat Systems Department, develops software configuration management support systems for use by the AEGIS Systems Engineering Division of the Naval Sea Systems Command, the largest program in the surface Navy. The complex AEGIS system is deployed on over fifty U.S. cruisers and destroyers, creating a major logistics and support challenge for the Navy. The original information tracking system was created in 1984 and was quickly outgrown by the configuration management challenges facing AEGIS program managers.

In 1991, a small team of software engineers was called upon to reengineer the aging system into a large relational database with nearly a thousand customers scattered across the country. Phased increments of the system were released over a multi-year time frame. To date, three subsystems and numerous upgrades have been delivered. The final subsystem is currently under development with a release planned for fall of 1999.

All software built in the last five years has been developed using Computer Aided Software Engineering (CASE) technology and structured Information Engineering (IE) philosophies. The development effort has also created a software engineering process and methodology that is gaining positive recognition from other organizations. (More information on CASE is provided in Section 3.)

The most recent development effort embraces client-server, distributed databases and graphical user interface (GUI) technology. The development team initiated efforts to apply key practices of the Software Engineering Institute (SEI) Capability Maturity Model (CMM) in order to improve process execution and quality. (More information is provided on SEI and CMM in Section 4, Implementing CMM Practices at the Project Level.)

2.1 PROJECT BACKGROUND

To facilitate the reader's understanding of this paper, an overview of the project upon which this paper is based is documented in the following paragraphs.

On 31 July 1997 the most recent system modernization, ACCESS Baseline (BL) 2 Computer Program (CP) Change Control (CC) software application was deployed. The system is hosted in a client-server environment utilizing a graphical windows interface and a distributed

Oracle relational database. A customized query environment that operates with *Business Objects*, a commercial off-the-shelf (COTS) ad-hoc query package, augments the ACCESS BL 2 application. Deployment of the client software is accomplished via the AEGIS Combat System (ACS) Configuration Management (CM) web site.

Development of ACCESS BL 2 followed a customized version of Oracle's *CASE*Method* incorporating rapid design and development techniques. The development team conducted Joint Application Development (JAD) sessions with cross sections of the AEGIS software customer community. Using job-specific business processes as a basis for system validation, the JAD sessions provided a forum for developers and customers to simultaneously review system requirements in the context of CM business practices, and to effectively resolve open issues. These sessions produced information critical to the production of accurate and effective database structures, module specifications, and reporting requirements.

In addition to the design and development improvements, customization of the data transition approach also produced dramatic results. Developers created a transition engine to efficiently migrate legacy data to the new data structures early in—and continuing throughout—the development lifecycle. The data transition approach benefited the project by providing the following:

- A data test-bed for use during application development
- A starting point for data validation by both the business and development communities
- An increased level of confidence regarding final data transition
- A shorter cutover time to effect the move from the legacy system to the new system

The team of software engineers, composed of Government personnel and multiple contractors under the leadership of the author, faced enormous challenges during the project. However, through teamwork, dedication, and constant attention to detail, these challenges were met. Through their vision, leadership, and unique technical expertise, the team substantially improved the development lifecycle methodology, reduced time to deliver, and improved the ultimate quality and customer acceptability of the final product. These improvements and lessons learned will be instrumental in the continued success of the ACCESS program as the next phase of ACCESS reengineering continues. Additionally, many of the advances in technology and architecture and the graphical interface standards established for ACCESS BL 2 will be used to accelerate the modernization and continued support of previously engineered phases of ACCESS.

2.2 PROJECT PURPOSE

The purpose of the project was to reengineer the existing CP BL 0 application (which provides CM lifecycle support to AEGIS computer programs) with an interface to the AEGIS ACCESS equipment and administration applications.

BL 2 supports the process of creating, reviewing, maintaining, and tracking the status and implementation of CP-related change requests, including Computer Program Change Requests (CPCRs), Interface Change Requests (ICRs), Specification Change Requests (SCs), and Film Label Change Requests (FLCRs). In addition, it supports the various Element Change Control Boards and the AEGIS Weapons System (AWS) Change Review Board (CRB) in development of board agendas, minutes, and status reports.

The technical goals and objectives of the project were as follows:

- Exceed the functionality of the current system
- Provide a method for the customer to respond quickly to new representations of data
- Establish an interface with ACCESS equipment and administration systems
- Improve the capability of establishing interfaces with other applications
- Allow business rules to be modified with data, rather than code, whenever possible
- Provide an easily maintainable and well-documented application
- Provide a stable product for the CP community
- Develop an intuitive and self-documenting application
- Reduce data redundancy
- Improve integrity of business data via automated validation checks
- Simplify retrieval of business information by providing more flexible querying capabilities
- Migrate the legacy system to a GUI/Oracle environment

Other goals and objectives included the following:

- Focus on customer business priorities
- Remove barriers to information

- Establish customer ownership
- Deliver short-term results
- Whenever possible, incorporate Level 2 practices as described in SEI's CMM

Although difficult to quantify, all goals and objectives were met.

2.3 PROJECT SCOPE AND SIZE

The customer community for the ACCESS Computer Programs application includes 400 customers located primarily at the Naval Surface Warfare Center, Dahlgren Laboratory (NSWCDL) site, with others at Bath, Maine; Pascagoula, Mississippi; and Port Hueneme, California.

The application was built using Oracle Version 7 with Windows 95 and consists of 223 tables, 1941 columns, 31 modules, 7 menus, 35 reports, and a common report interface.

In its entirety, the project required over seven years. However, during the first four and one-half years, the effort struggled because of other, higher-priority, development efforts; inexperienced project leadership; personnel turnover; budget constraints; inexperience with a new release of the software development tools; conversion to a different development methodology; and customers' redefinition of business processes.

As many as 17 people were employed during the effort. It is beyond the scope of this paper to provide the aggregate cost of this effort.

The software development methodology is discussed in detail in Section 3, Lifecycle Development Methodology; however, some preliminary understanding of the major events and milestones will enhance the reader's understanding of the magnitude of the effort. Figure 2-1 details the major events of the development effort from the time the author assumed responsibility as project leader for the development effort. The timeline does not include strategy of other analysis tasks.

Within each stage of development are milestone events that ensure the integrity of the software. The following list indicates these major events for each stage.

Analysis:

Preliminary Design Review

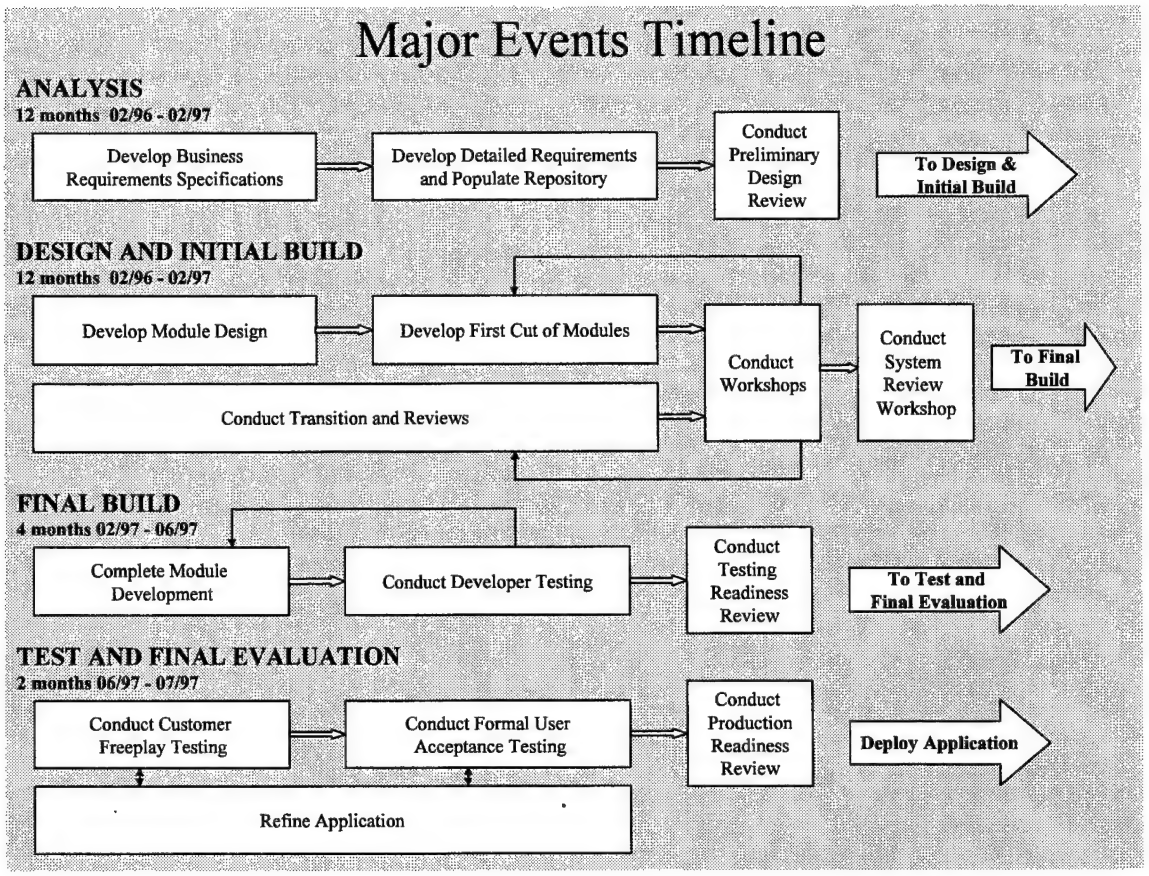

FIGURE 2-1. MAJOR EVENTS TIMELINE

Design and Initial Build:

- Database Design Review
- Data Transition Review
- Customer Workshops
- System Review Workshop
- Developer and Peer Reviews

(Note: The Customer and System Review workshops equate to Critical Design Reviews)

Final Build:

- Developer and Peer Reviews
- Test Readiness Review

Test and Evaluation:

- Freeplay Test Reviews
- User Acceptance Test Reviews
- Production Readiness Reviews

The activities listed above are structured activities that require significant time to plan and conduct. In this effort, the team was responsible for these major events.

The major milestones are the completion of each of the development phases. In addition to the magnitude of the development work, there are many other tasks that must be conducted in parallel to the development work. The project leader may or may not be in charge of each of these support tasks. In this effort, the majority of support tasks were managed by the project leader and supported by the development team, increasing the complexity of the management effort and the workload on the developers.

Some of the major tasks and the designated responsibilities in this development effort include the following:

Under the responsibility of the project leader:

- Transition, database design reviews, transition reviews, and data quality reviews
- Common report interface development
- Synchronization between the portion of the old system not yet developed and the new application
- Cutover from the old system to the new system
- Ad-hoc reporting development
- Client-Server deployment strategy and mechanism (support task)
- Test plans and procedures planning and development (support task)
- User Guide development (support task)
- Customer training (support task)
- Customer testing (support task)

Outside of the responsibility of the project leader, but requiring coordination:

- User enrollment to new system
- Server readiness
- Customer support readiness

2.4 PROJECT CONSTRAINTS

As mentioned above, the project was experiencing difficulties when the author accepted responsibility as project leader. Immediate steps were taken to get the project on track.

Given the longevity of the project, almost all development aspects changed, including methodology, architecture, development tools, etc. As a result, the original development team had neither the skills nor, more importantly, the experience necessary to effectively complete the effort. Project funding was extremely tight, and the budget did not support retraining or hiring additional experience. The project leader was able to convince the customer of the seriousness of the problems, and the customer provided additional funding for hiring skilled talent from other contractors. This action, however, added complexity to budget planning and execution.

The new team members provided a solid GUI, client-server development foundation, and facilitated the training of current team members. Consultants were hired to help put the methodology in place and to provide expertise in developing processes to support the methodology. Although the project leader had the necessary education to know what to do, the consultants provided the "how to do" expertise.

Another constraint was the mix of developers from many contractors, which further complicated management of the effort. However, the mix of developers brought the much-needed skill set and experience required to do the job. Late in the development effort, a major contract change almost destroyed the carefully built team of experts. However, through the concerted efforts of the project leader, management, the customer, and the winning contractor, the team was kept intact. Some time was lost because of the contract change, but the time lost was minimal in comparison with the time required for a mostly new team to become productive.

Matrix management imposed additional constraints and management complexity. The project leader reported to one chain of command for direction and another for the customer requirements. Over time, this arrangement improved as the two chains of command, for the project and for the customer, improved communications.

Resource limitations were caused by inadequate funding and necessitated using developers to accomplish the required support activities. Leadership skill and finesse were needed to motivate the team members to accept these additional responsibilities, such as developing test scenarios and training customers.

Because the iterative development approach was not well understood, it was not initially well received by management. Iterative development is almost the direct opposite of structured approaches used for developing tactical systems. Management acceptance was gained by an approach of continual review of iterative development techniques. To date, the methodology is still questioned by many managers; however, the successful deployment of the application has done much to obtain support and establish the methodology.

2.5 PROJECT TECHNOLOGY

The following list depicts the key technologies used in the application:

Oracle 7 Relational Database

- Server: Oracle version 7.3 on Vax4000 running UNIX
- Normalized relational model using American National Standards Institute (ANSI) standard Structured Query Language 2 (SQL2)
- Use of stored procedures and triggers implemented at the server controls row-level security
- Role-based security privileges

Client-Server Application

- Distributed environment using SQL*Net 2.1 Transmission Control Protocol/Internet Protocol (TCP/IP) to provide connectivity between client workstation and database servers
- Oracle Forms 4.5 and Reports 2.0 (GUI) running on the client-server

CASE

- Designer 2000 application development environment
- JAD/Rapid Application Development (RAD) methodologies and techniques
- Generated Developer 2000 modules

Other COTS Software Tools

- Production Version Control System (PVCS) for release management of software components for multiple development and production releases
- Business Objects for ad-hoc querying and reporting capability
- Software Quality Assurance (SQA) for tracking of defects and automated testing procedures

SECTION 3

LIFECYCLE DEVELOPMENT METHODOLOGY

Two documented processes, the Oracle *CASE*Method*, described in Reference 1, and the Oracle *CASE*Method for Fast Track Development*, described in Reference 2, evolved to form the lifecycle development methodology for this project. The author and the development team members facilitated the evolution of the Fast Track Method. The evolution of the current development methodology is described below.

3.1 METHODOLOGY EVOLUTION

The project methodology is based on the original foundation of previous software projects developed using *CASE*Method* by Oracle, which is "a structured approach to carrying out the total task of developing systems to meet business needs." This methodology is similar to many waterfall methodologies, and the "tasks are grouped into major stages, each with clearly defined deliverables, that are necessary for subsequent stages." *CASE*Method* provides a framework for the lifecycle of the business system. The steps in this methodology include Strategy, Analysis, and Design, followed by parallel efforts of Build and Customer Documentation, concluding with Transition and Production.

In the mid-1990s, Oracle applied the original principles of the methodology and modernized these principles as the *CASE*Method* for Fast Track development, which is based on RAD. According to Barker, in Reference 2,

> As the systems development methods have matured and their use has become more widespread, the technology for automating the development process has matured. Initially, CASE tools were developed to automate the processes as defined in systems development methods. But the technology quickly reached a stage where it is possible to challenge the existing methods and make new approaches possible. CASE tools now offer the prospect of re-engineering the business of information systems development. They offer the opportunity to make the transition to the next stage in the process of formalizing information systems development. The fast track approach places the emphasis firmly on exploiting CASE for productivity, without sacrificing the risk reduction achieved with structured methods.

The fast track approach builds on the success of CASE method in defining the lifecycle and developing for a deliverables-oriented approach. The same basic lifecycle remains, as do many of the tasks. However, fast track modifies the lifecycle to exploit alternative techniques that use evolutionary iterations, often known as "iterative builds" to produce deliveries.

Another major difference in fast track is the project management approach. Fast- track adopts a more flexible strategy that is best described as risk adaptive as opposed to risk aversive. There is intrinsically more risk in the evolutionary approach. It is impossible to know for certain in advance how many iterations will be necessary, but provided the number of iterations is low, the exploitation of faster techniques means a net gain.

The third departure is in customer involvement. In fast track projects, the level of customer involvement stays high throughout the lifecycle of development. This is because customers are, in effect, participating in acceptance testing from start to finish.

3.2 PROJECT LIFECYCLE DEVELOPMENT METHODOLOGY DETAILED DESCRIPTION

As shown in Figure 3-1, the project lifecycle development methodology is an adaptation of the *CASE*METHOD* fast track customized to support the N27 development environment. (Section 3.5 describes the detailed tasks of the methodology.) A box is drawn around Analysis and Design to emphasize that all requirements are not defined to the lowest detail prior to Design. During Design, the customer, through a series of JAD sessions known as Module Design Workshops, refines the requirements until they are complete by participating and experiencing the software as it develops. (Workshops are detailed in Section 4.4.)

Also, note that the Design and Initial Build tasks are in the same box to indicate that these activities are an iterative process. A first cut of software, without a lot of functionality, is built based on a preliminary design. The requirements are refined, the design reflects those changes, and the software goes through another build iteration. By the time the software enters the final build stage, what is left to complete is underlying code at the database and library levels to implement security, special procedures, integrity, etc. The "guts" of the system are completed to allow the modules to function in compliance with the customer's requirements.

Another deviation from the traditional methods is the early transition effort. Transition is actually split into two tasks. The first task concentrates on having translated data available when the module design workshops begin. This allows the customer to have real and current data available for viewing the screens and the reports. The second part of the transition is continual refinement and validation of the data throughout the workshops and initial build. The goal is to have transitioned data that is accurate long before formal testing begins. Another enhancement is

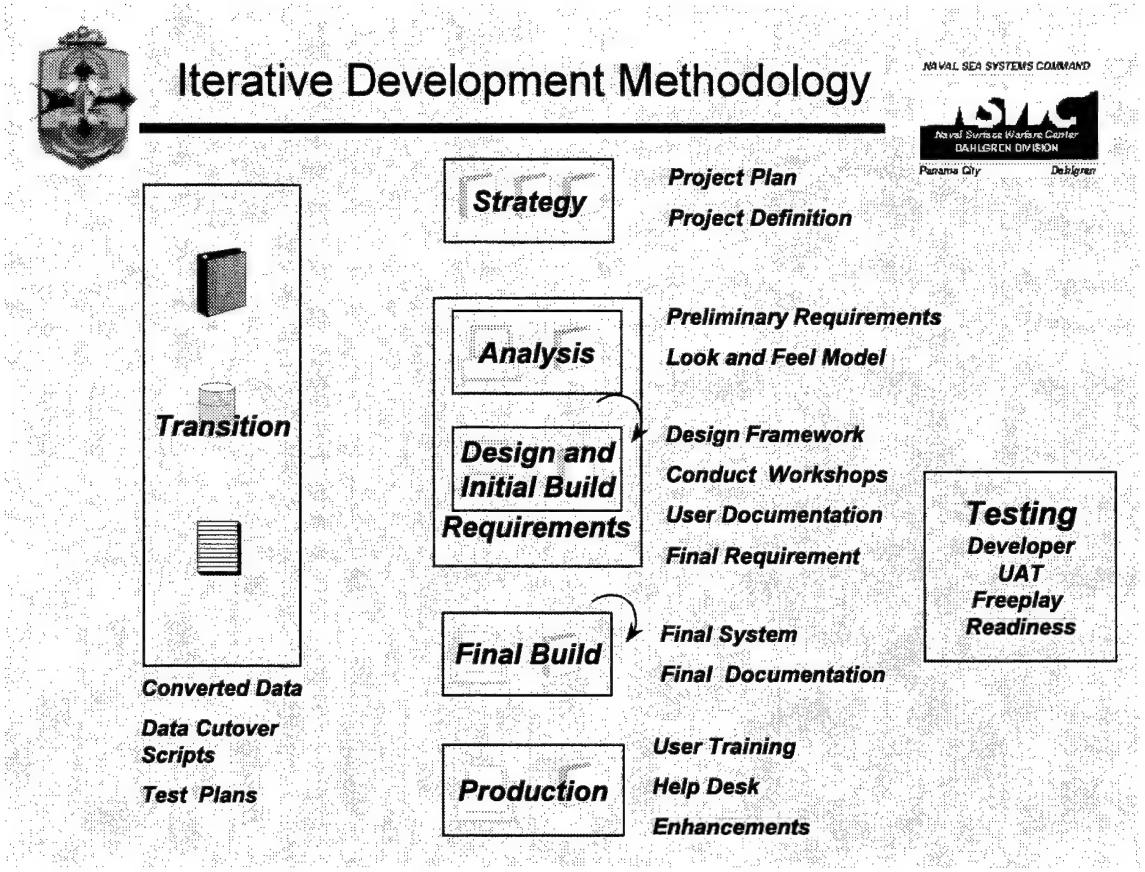

FIGURE 3-1. LIFECYCLE DEVELOPMENT METHODOLOGY

the development of a transition engine that takes data from the old system and, through programmatic scripts, develops a process that can repeatedly take the old data and translate it into the new data structure. (Transition is discussed in more detail in Section 3.5.4.)

Another area that is tailored to support the project methodology is testing. Testing is supported early in the development process and there are a number of different types of tests that continue throughout development. Testing begins in Initial Build with informal developer, peer, and limited customer testing. During Final Build, there is formal user acceptance testing (UAT), freeplay testing, and just prior to deployment, readiness testing. (Testing is discussed in more detail in Section 4.4.)

3.3 CUSTOMER INVOLVEMENT

Customer involvement is considered crucial to rapid development. Involvement begins early, is most intensive during workshops, and continues until the project is ready to deploy. This project expanded customer involvement to include a full-time business liaison person, who was assigned to the development team and who worked directly with the project manager and the

developers on a daily basis. The list that follows documents some of the major contributions made by this person:

- Discussed design issues and questions at regular customer meetings; worked with the manager of Computer Programs Software Configuration to resolve issues that could not be resolved through regular business activities

- Screened all customer-requested design changes prior to discussing the changes with the developers

- Translated at all workshops and meetings when customers and developers had difficulty communicating

- Assured good attendance at workshops by selecting dates and times when the customer community was most available

- Identified and notified participants of upcoming workshops and meetings; followed-up prior to each meeting or workshop with a reminder, and when necessary, delivered materials to the participants prior to meetings and workshops

- Produced all the user acceptance test scenarios for formal testing to ensure all business functionality and all customer roles were addressed

- Worked with the development team to pretest software prior to customer testing, which ensured the customer did not spend time testing software that was not ready

- Worked closely with the project leader on planning and scheduling and providing background on procedures and politics

- Created a customer's procedure manual that documented how to use the software application to support business processes, such as complex change control boards

- Arranged for a state-of-the-art training facility to be ready and available when testing and training were required

- Brought confidence to the customer community to enhance the customers' reception of software with functionality much different than that to which they were accustomed

3.4 TRAINING

Customer training is not discussed in detail in the methodology; however, for a customer to be able to satisfactorily use complex software, training is imperative. The original plan was to train approximately a dozen key users who would, in turn, train the customer community. This

approach is a common tactic planned for many development efforts, but it is seldom effective. In this effort, after an assessment of the workload and the extensive time required to train these key customers to use complex software, the original plan was abandoned.

It became apparent that the initial training would require the combination of the project leader, key developers, and the customer liaison. This approach proved to be effective. After the initial deployment effort, the customer liaison conducted all follow-on training.

3.5 DETAILED TASKS WITHIN THE METHODOLOGY

The following section documents the detailed tasks within the methodology. The different stages in the lifecycle development are defined followed by a listing of tasks in the stage and other pertinent information. The focus of this paper is on iterative development. For completeness, only a cursory overview is presented of the Strategy phase. It is assumed that necessary planning and high-level analysis will be conducted prior to when the project enters the iterative phase of detailed Requirements Definition, Design and Initial Build. For ease in presentation, Transition is discussed after Final Build, but in reality, Transition is parallel with the Requirements stage.

3.5.1 Strategy

The objective of the fast-track strategy or planning stage is to obtain the following:

- A clear definition of the project's scope
- Prioritization within the scope
- The plan for the project

At this early point in the development lifecycle, the Project Plan is understood to be a visual representation of major phases that must be completed prior to delivery of the software. The plan is not intended to provide accurate time periods for tasks or a mechanism on which final delivery schedules should be based.

A sample plan of action using the lifecycle development methodology can be found in Appendix A.

The components of the Project Plan should include the following:

- Resources, people, equipment, and facilities
- Description of the deliverables (such as prototype, test plans, executable software, customer documentation)

- Tasks required for each stage to produce the deliverables
- Estimated task duration and how resources will be scheduled to accomplish the tasks
- Major milestones that can be used in tracking the progress of the project
- Configuration management and quality assurance processes that will be applied to ensure deliverables meet the expected quality

For this effort, the following process improvement was added to the fast-track approach:

The Statement of Work (SOW) is one of the process improvements incorporated in this effort. In this particular project, the SOW was not produced until the Analysis stage, at which time the author accepted responsibility as project leader. However, this project leader felt the document was needed to facilitate an understanding between the customer and development communities, and therefore the SOW was produced and agreed upon near the end of Analysis. The SOW adds an extra level of commitment and understanding to support the Project Plan. The SOW (in Appendix B) is discussed in detail in Section 4.3, Software Project Improvement Implementation for Software Project Planning.

3.5.2 Requirements (Design and Build)

Analysis is the first task of the Requirements stage. During Analysis, findings from the Strategy stage are expanded into a set of verified preliminary requirements. An initial GUI "Look and Feel" Model is developed to assist the customer in understanding how the software will appear and function. This model or prototype does not address how the software will function to support the requirements. The intent of the prototype is to help the customer visualize how the new application will function and to lay the groundwork for visually defining requirements. This stage should culminate in a System Requirements Review, which demonstrates that the requirements are balanced with the original project scope, cost, schedule, and risk.

Major tasks of the analysis include the following (note that additions to the fast-track approach are annotated as process improvements to the applicable task):

- Review requirements for clarity
- Prepare and conduct interviews to obtain additional details
- Construct a Requirements Specification Hierarchy (process improvement: see Appendix C for an example)
- Establish a preliminary design, including an entity relation diagram and functional model

- Document analysis findings in the CASE tools
- Develop prototype
- Prepare and conduct a System Requirements Review (process improvement)

Deliverables of the Analysis stage include:

- Look and Feel Model
- Requirements Specification Hierarchy (process improvement)
- Database Design
- System Requirements Review Documentation (i.e., business flowcharts, module design approach, known issues, etc.)

Once a preliminary set of requirements is developed, the iterative nature of the process begins. During Design and Initial Build, the software is developed. Initially, the software has the minimal functionality needed for the customer to verify and refine the requirements before complex development is undertaken. Through a series of structured workshops, customer input is received to validating the design review. This information is documented in the workshop checksheets and other supporting documents and managed by using several tracking tools, and the software is expanded to include the input. The goal of the workshops is to ensure the requirements have been adequately incorporated into the software design and to receive a formal, signed customer acceptance of the design.

During this phase, developer and peer testing are conducted. This testing should be structured, documented, and managed. (Additional details about testing can be found in Section 4.4.)

It is imperative that the system design be frozen early in Design and Initial Build to avoid "scope creep." The customer must commit to this requirement and actively seek ways to limit changes. Occasionally, there will be a requirement to make a design change when implementation does not effectively support the requirement, but design changes should be minimized and infrequent. One approach that worked for this project was officially adopting formal change control procedures that included logging and formal review of these changes. The changes were reviewed by a board consisting of developers and customers to determine which changes were needed in the initial release and which could be deferred. Negotiations on schedule impact were conducted.

There were many workarounds, often known as requirement waivers, authorized to allow the design to remain stable, but these workarounds made the software less than optimal. After the initial release of the software, these changes were reviewed and those with the highest priority were implemented in subsequent releases. The important point is that sometimes

functionality must be compromised in order to deliver a timely product; however, functionality should be implemented at the earliest opportunity.

A trend that appeared early in the development lifecycle was a customer not liking the way the software worked and logging an enhancement request. After training and initial use of the software, the customer often found the concern was a matter of education: Familiarity with the software eliminated the problem, and the change was withdrawn. The moral of this trend is to persuade the customer to wait until he or she is familiar with the software before judging its ease of use.

Major tasks of Design and Build include:

- Create and modify the database structure
- Develop module designs for query and maintenance screens and reports
- Develop detailed module specifications and business process models
- Prepare and conduct functional design workshops
- Develop modules with limited functionality
- Conduct developer and peer software evaluations

Deliverables of the Design and Initial Build activities include:

- Documentation and results of the functional design workshops (See Section 4.4.5 for details on how improved processes were developed to support this task.)
- Business Process Model (See Figure 3-2 for an example.)
- System Architecture Model
- Software that demonstrates all requirements are in the design
- Software that has been developer- and peer-evaluated

3.5.3 Final Build

During Final Build, the core requirements of the system are developed. This includes database-level development, such as implementation of security, building common libraries, and adding the capability to make the software function and perform to support the requirements. As

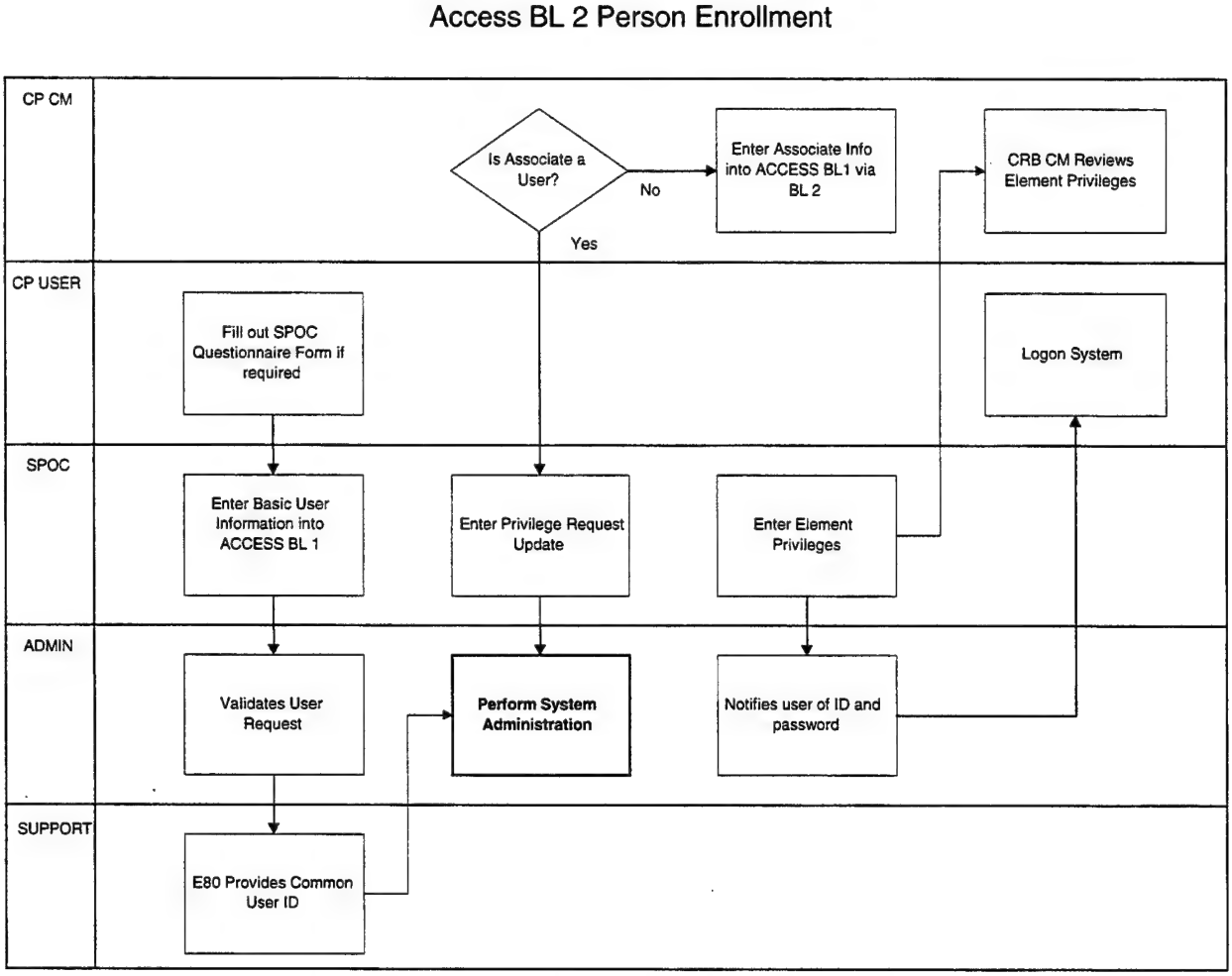

FIGURE 3-2. SAMPLE BUSINESS PROCESS FLOW DIAGRAM

much as possible, CASE tools and code generators are used to produce the system. Customers remain an active part of this phase and should frequent the lab to assist in testing of the software. The final activity of this phase includes a comprehensive workshop demonstrating the entire functionality of the software.

Major tasks of Final Build include:

- Develop fully functional modules with all required special processing
- Conduct developer and peer testing
- Prepare and conduct system integration workshops

Deliverables of Final Build include:

- Code and implemented database
- User documentation
- Test plans and other criteria to support acceptance testing
- Training plans
- System development documentation
- Signed customer acceptance of the system design and functionality

3.5.4 Accelerated Transition

As shown in the lifecycle development diagram (Figure 3-1), transition starts early in the development lifecycle. During Analysis and Design, the software transition engine is designed and built and the data model and transition mapping accuracy is validated. The accelerated transition process is focused on reducing overall project risk by presenting end products to the customers earlier and at a lower cost than expected.

Traditionally, legacy data is presented to customers in small data sets toward the end of the Final Build phase. Discrepancies in the database design can be masked by review processes that focus on the collection of data in screens and reports using only small samples of data, often produced by hand. With the accelerated method, transitioned data that does not support the data presentation needs of the customer community is recognized early in the lifecycle of development.

The goals of accelerated transition are:

- Immediate insight into the business rules
- Real customer involvement in transition
- Early ability to validate the data model
- Use of operational and current data in testing
- Empowerment of timely data cleanup efforts
- Early availability for use with ad-hoc COTS products

- Minimization of build changes
- Risk reduction

The transition process used in this effort is documented in Figure 3-3. For the transition process, the team performed the following:

- Identified the data to be transitioned from one baseline to another
- Developed and loaded database tables with preload values (new data) and translated values (changes to existing data)
- Developed and loaded a rules database, which has hundreds of rules for how the data is to be transitioned
- Developed the transition engine, which contains the programmatic scripts that take the data from the original baseline and transition the data with the preload values, the translated values, and the rules to the new database

Once the data is transitioned, it is available to be used in ad-hoc reporting and for testing the modules and reports.

Transition was an activity planned to parallel the development of the software. In this project an accelerated transition was initiated, but because of resource problems and a major contract change, the transition did not stay in sync with the development and problems resulted. Data errors were not found until later in the build effort, which required reprogramming of the transition engine, additional transitions (which took a great deal of computer and developer time), and revalidation of data. Because the customers did not have complete, realistic, and adequate data to use for testing, they had a more difficult time validating that the modules and reports were correct.

In addition to impacting the quality of the development and testing, using poorly transitioned data had a negative impact in customer confidence. Fortunately, intense efforts of knowledgeable team members enabled the transition of data very successfully with only three minor deficiencies. However, the workload on these people, at an already demanding time, affected other areas of the project and put a tremendous strain and workload on already overburdened people. These lessons-learned re-enforced the need for accelerated transition.

3.5.5 Completing the Lifecycle

During this phase, the system moves from development to production. Plans were put in place for the cutover of the system. Several types of testing were completed. Customer acceptance procedures were developed for detailed module testing and system integration testing.

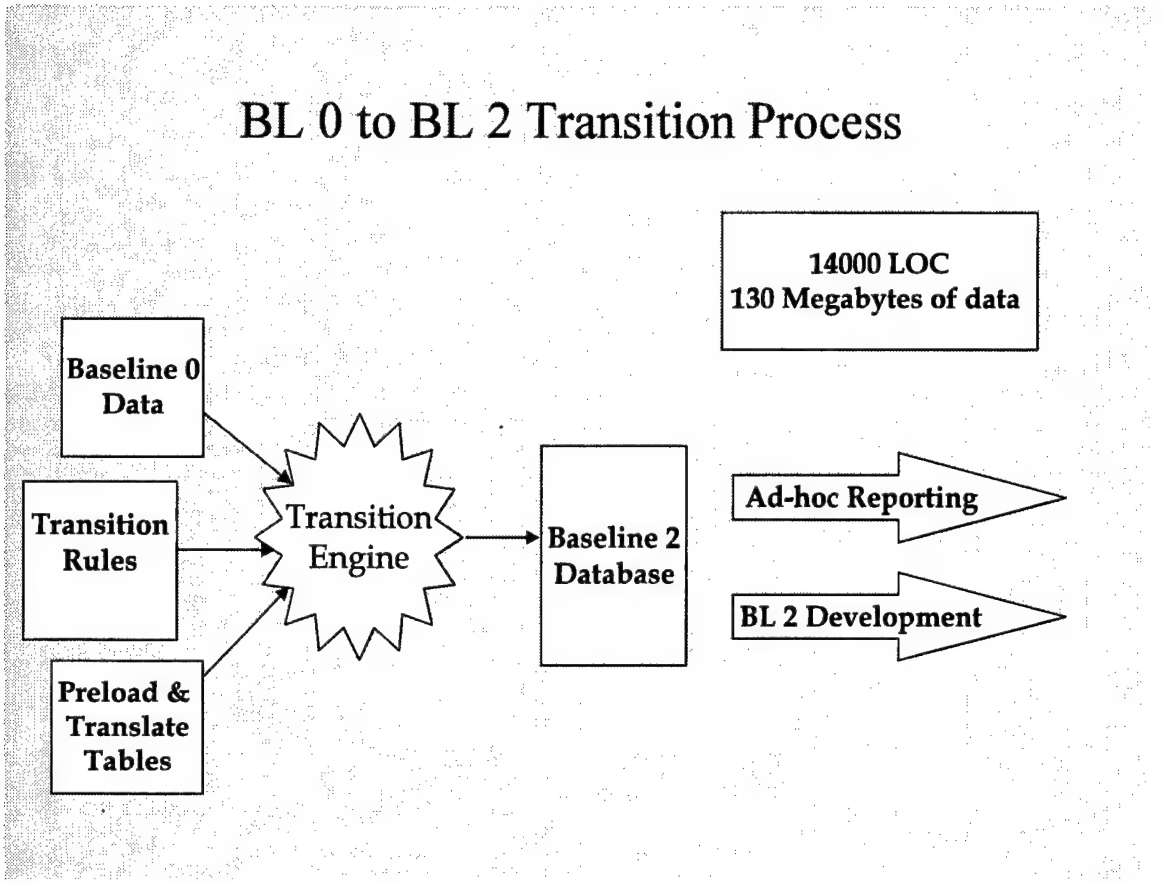

FIGURE 3-3. TRANSITION PROCESS

(Testing is covered in more detail in Section 4.4.) A period of freeplay, unstructured testing was conducted. A final readiness testing was completed as part of the deployment process. Customer training was conducted. The customer support service was readied.

During the final cutover, the final data transition, database and application installation was conducted. Cutover for this application was a complex set of tasks that sometimes ran in parallel to reduce the customer downtime, but often required task execution in a serial fashion. The cutover period began on a Thursday afternoon and was completed the following Tuesday morning, right on schedule. Two extra days were added to the schedule to allow reruns of portions if necessary, and to accommodate the expected downtime caused by electrical storms of a typical Virginia July. Developers and database administrators worked in shifts to minimize the downtime. Check points were built into the process so that no more than a day would be lost.

Some of the tasks during the final cutover included the following:

- Freezing the original system
- Doing backups and reports from the old system

- Downloading the data
- Transitioning the data
- Loading the data
- Loading the new database and applying the necessary database links
- Running and validating scripts and reports to ensure integrity
- Compiling the application
- Packaging the application
- Distributing the new application
- Enrolling the users

After the cutover tasks were complete, the new application was made available to the customer community.

Part of completing the lifecycle is follow-up with the customer after delivery, assessing their satisfaction with the software.

SECTION 4

IMPLEMENTING CMM PRACTICES AT THE PROJECT LEVEL

The development of software continues to increase in complexity. Other disciplines of science and engineering have found successful methods for managing the quality of their products, such as in the development of hardware. Yet managing the development and quality of software continues to be an elusive and frustrating challenge. During the last few years, new methods have been introduced to help the manager and the developer in tackling this ever-increasing problem. One of these techniques is the Capability Maturity Model, which helps an organization improve their processes in developing high quality software.

This section addresses ways in which implementation of SEI CMM key practices at the project level can be beneficial. Carl Dicter states in Reference 3, "Documenting your software-development process is the only way to ensure that it is complete, repeatable, and improvable. Fail to do so and you are likely to remain in a chaotic development mode. A properly documented software development process can improve quality, save time and make work more enjoyable." The implementation of previously defined processes improves the ability to deliver complete and quality software products. Work is more successful and enjoyable when it is not chaotic, people know what to do, and when and how to do it.

4.1 BACKGROUND

Several years ago, the SEI of Carnegie Mellon University developed a process called the CMM. Several departments at NSWCDL have begun initiatives to apply these principles to software development. The adoption and institutionalization of the CMM by the organization requires commitment at all levels of the organization, investment in resources to support the discipline, and a significant time frame for realization of improvement in the software development process.

Many of the principles upon which the CMM is built can be used at the individual project level with significant results. This section addresses how a select group of processes were developed and implemented and describes the resulting improvement in the quality of the software being developed. It is this project leader's belief that the best approach is to establish the CMM throughout the organization; however, immediate improvements can be made at all levels by applying the SEI principles.

The improvement of software quality is difficult because organizations tend to lock on to a specific pattern of doing things. They adapt to the current level of quality and do not understand what is required to change to a new level. Since they do not know, they do not try to change. The practices described in this document assist the software practitioner in understanding what to change and how to make the change happen. The enlightened developer or manager can take the SEI CMM to levels below the organization, apply the principles, and benefit from the resulting improvement in quality. Even a small amount of process improvement at the project level will net a large return for the initial investment.

At first glance, many activities may appear obvious as a project requirement, but in reality, many of the basic principles needed for the development of quality software are often overlooked. An organization need not wait until every process is documented and a plan is developed to improve the development process before experiencing improvements. Application of quality improvement principles can be applied successfully to any task, regardless of size, for improvement of quality of the product developed. The development of software can be improved even as the technical complexities of the software increase.

The development and institutionalization of the review process incorporated in this development effort improved the quality of the software under development. It is important to note that hundreds of errors were detected early in the lifecycle of development when the errors were easy to correct and of little financial impact. Once developers thought they were no longer being singled out for making errors, a team bonding formed, with members actively working together to improve the quality of the software. Team members began to approach developing software differently by learning from errors found in the past. The developers began to openly interact with each other in critiquing the work.

This discussion focuses on the processes developed and lessons learned for specific key practice areas at the project level. Before proceeding with details of the implementation using key practices, a brief understanding of the SEI CMM is provided.

4.2 THE SEI CMM

Watts Humphreys, in Reference 4, states in the part titled "The Software Process Capability Maturity Model" that "CMM deals with the capability of software organizations to consistently and predictably produce high quality products. It is closely related to such topics as software process, quality management and process improvement." The CMM is a framework intended to help organizations assess the current level of capability of the software process, discover strengths, weaknesses, and risks in the software process, and select the most important areas for improvement. Humphreys notes that to improve software capabilities, organizations must take five steps.

1. Understand the current status of their development process(es)

2. Develop a vision of the desired process

3. Establish a list of required process improvement actions in order of priority

4. Produce a plan to accomplish these actions

5. Commit the resources to execute the plan

As shown in Figure 4-1, the steps of the software process are classified as one of five maturity levels:

- Initial
- Repeatable
- Defined
- Managed
- Optimized

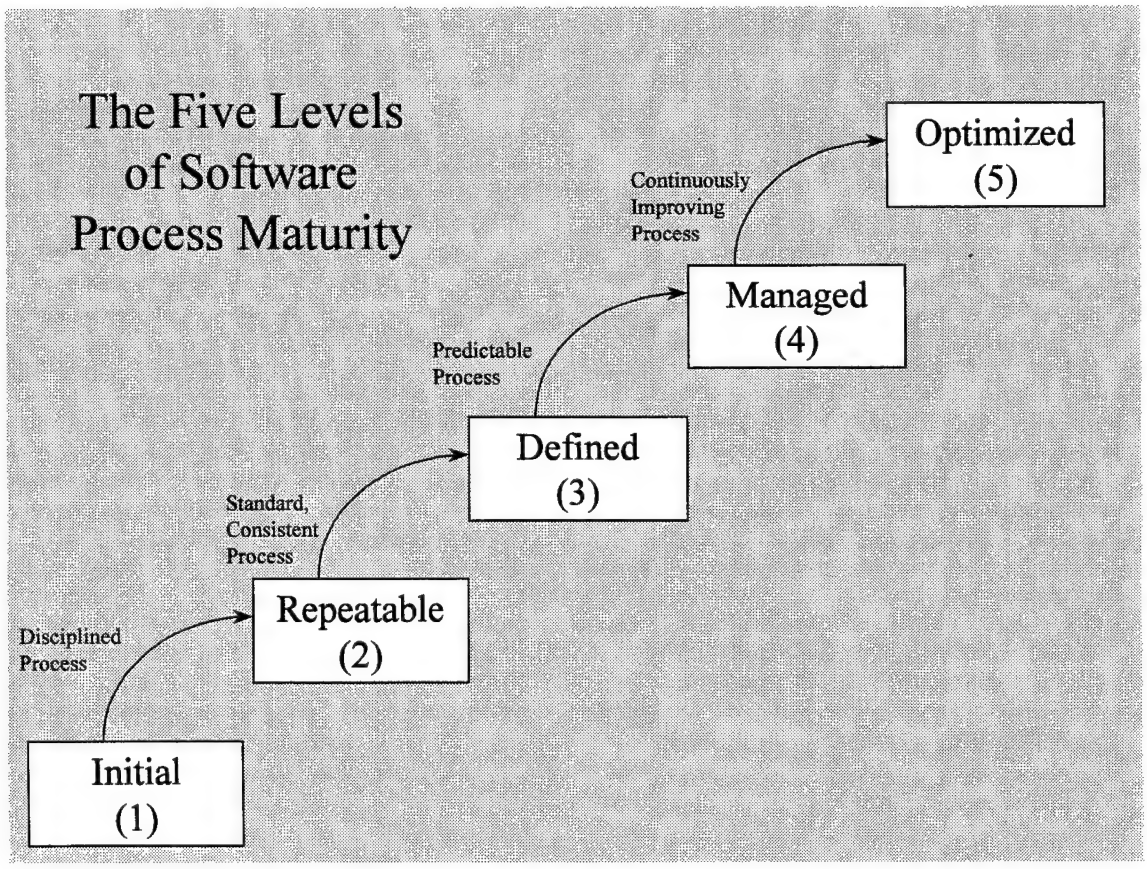

FIGURE 4-1. CAPABILITY MATURITY MODEL

By comparing their organizations' practices and capabilities to benchmarks, software organizations can identify areas where improvement actions are most likely to produce a basis for sustainable improvements in capability. Each maturity level represents a level of process capability and each level consists of a set of key processes, as shown in Figure 4-2. CMM Key Process Areas (KPAs) are established to achieve goals. KPAs are organized by common features, which address implementation or institutionalization of key practices. The key practices are described in Reference 5. Level 2 of Software Process Maturity, applied to this development effort, is characterized by being repeatable.

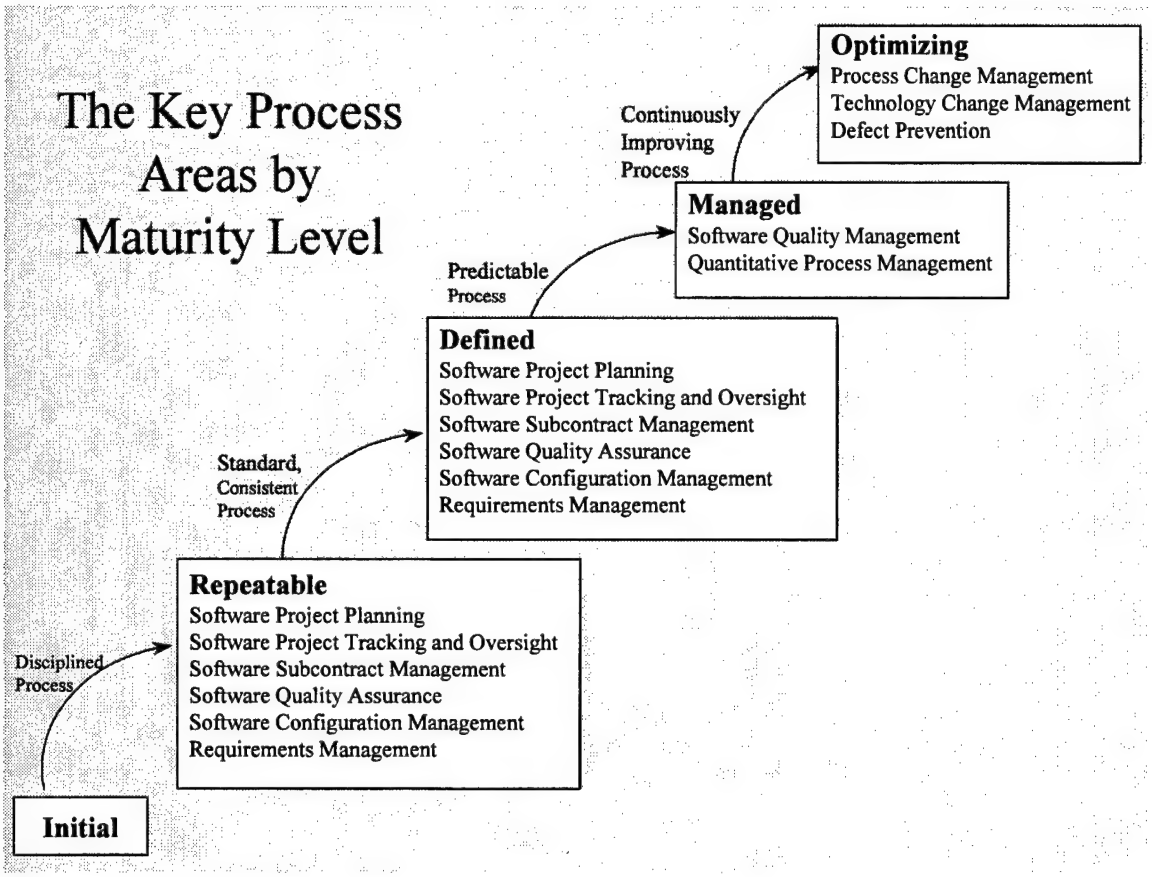

FIGURE 4-2. CMM KEY PROCESS AREAS

Basic project management processes are established to track cost, schedule, and functionality. Paulk describes, in Reference 6, the necessary process disciplines needed to achieve successes with these basic management processes. The repeatable key practices from Level 2, implemented in this development effort, include Software Project Planning and Software Quality Assurance. These practices are documented in the following sections.

4.3 SOFTWARE PROJECT IMPROVEMENT IMPLEMENTATION FOR SOFTWARE PROJECT PLANNING

The purpose of Software Project Planning is to establish reasonable plans for performing the software engineering and for managing the software project. Software Project Planning involves developing estimates for the work to be performed, establishing the necessary commitments, and defining the plan to perform the work. These estimates are documented and used in planning and tracking the software project. For this project, we established a Software Project Planning improvement goal to create an SOW to define scope and thus more consistently and accurately manage the planning aspect of the effort.

Software development includes documenting constraints and goals and defining and bounding the software project. The software planning process includes steps to estimate the size of the software work products, the resources needed, a schedule, identification and assessment of software risks, and negotiated commitments. In Reference 7, Olson describes the steps necessary to establish the plan for the software project.

The goal, to develop an effective Software Project Planning process in accordance with the tenets of CMM Level 2, was expected to achieve the following results:

- Software development estimates are documented for use in planning and tracking the software project.
- Software project activities and commitments are planned and documented.
- Affected groups and individuals agree to their commitments related to the software project.
- Project management is effective and satisfactorily complies with the project's mission.

4.3.1 Statement of Work

One of the process improvements incorporated in the development methodology to improve project planning execution and management was the development of the SOW. The SOW consisted of the following sections:

- Purpose
- Scope of the work
- Technical goals and objectives
- Other support goals and objectives

- Identification of customers
- Imposed standards
- Assigned responsibilities
- Cost and schedule constraints and goals
- Dependencies between the software project and other organizations
- Resource requirements
- Resource constraints and goals
- Critical success factors

The SOW need not be excessive in length. The goal is to provide a means of communication between management, the development staff, and the customers. (The SOW for this project is included in Appendix B.)

The development and acceptance of the SOW generated an understanding between all parties. Prior to the development of the SOW, there were several misrepresentations of the scope of the project, the constraints, and the goals. The development staff and the customers were able to identify these misunderstandings and reach a consensus. The document continued to be used as a vehicle of clarification and communication throughout the development effort. A side benefit of developing the SOW was enhanced customer understanding of the effort and buy-in to the development process.

4.3.2 Project Risks and Contingencies

Another of the Software Project Planning activities deals with risk. The risks associated with the cost, resources, schedule, and technical aspects of the project are identified, assessed, and documented. The risks are prioritized based on their potential impact to the project. As shown in Reference 8, contingencies for risks are identified (examples of contingencies include schedule buffers, alternate staffing plans, and alternate plans for additional computing equipment).

Categories of risk should include, but are not limited to cost, resources, schedules, staffing, and technical. To achieve the goal of improved risk management, a Project Risk and Contingency Plan developed early in the development effort is attached as Appendix D and provides examples of risks and contingencies.

Prior to the development of this document, there was no formal process for documenting risks or for providing management with the information necessary to develop contingency

approaches for dealing with the risks. The project leader, working with the team, developed a repeated process for risk contingency and planning. Project risks were identified and documented and presented to management. Management became more aware of and in tune to the needs of the project. An early awareness of concerns provided an opportunity for early problem resolution. Problems were addressed before they exploded and became difficult to resolve. Another benefit is the ability to see the combined impact of all risks. Previously, a risk might have been noted and reported, but not in the context of all the other risks, so the severity of the combination of risks was not visible.

One of the best ways to identify risks is in a development team brainstorming session. After analysis, but before design, a brainstorming session was conducted by the developers, database administrators, system administrators, and selected customers. The risks were classified as either a project risk or an open related issue. The risks identified from this effort were added to the Risk and Contingency Plan.

Often only one person was aware of a concern that had not previously been documented. Such concerns were valid and added to the document. One example is that none of the developers knew the new operating system software well enough to realize that E-mail capabilities were going to be a problem. The networking guru assumed everyone knew about the system's limitations. As a result of this interaction, this previously unidentified issue was worked earlier in the project rather than closer to implementation when resolution would have been more difficult and costly.

Another advantage of the brainstorming session is that one person's comment will suggest another possible risk that might not have been previously recognized. Team members heard a risk and made another association. When brainstorming sessions are conducted, it is suggested that a recorder be present, who will, at a minimum, document the following information:

- Issue description
- Primary responsibility
- Affected groups and tasks
- Risk assessment

All of the issues identified may not be critical or be included in the risk contingency document, but going through this process minimizes the risks that might be overlooked.

From a project management perspective, the project manager is able to share concerns with higher level management when the problems first need to be addressed. The project manager is more confident of success when the understanding and resolution of problems is addressed by all necessary parties. It should be noted that for this process to work, an openness and understanding of the process must be shared. The project manager must take care not to hide

or diminish the severity of the suggested risks. Management must be careful not to assign blame when problems are identified.

Another technique that can be used with risk projection is level of confidence in estimated risks. For example, early in the lifecycle of development, the risk is often the greatest because of many unknowns and variables. As the project nears completion, risk level can be projected more accurately because there are fewer unknowns, and the project is more stable. After each major milestone, the risk in prediction should be less. In Table 4-1, the project leader is only 50% sure, after high-level analysis is completed, the original projected delivery date will be met. After completion of each major task the delivery date may require modification.

TABLE 4-1. EXAMPLES OF RISK-LEVEL CONFIDENCE

Task	**Level of Confidence in Project Completion Date**
High Level Analysis completed	50%
Detailed Analysis completed	60%
Design Completed	75%
Initial Build Completed	85%
Testing Completed	90%
Schedule Revision Points	
Design Review Completed	02/06/97
Module Redesign Completed	03/05/97
Prototype presented and Accepted	04/1/97

The important point to be made is that the project manager must be honest with management that a firm date cannot be projected; however, the confidence in the delivery date will increase as the project progresses. Over time, this project manager found that higher-level management, although sometimes frustrated with the vagueness of the scheduled completion date, was more receptive to projections of confidence based on the level of risk for tasks than to wild-guess projections that were continually modified. It may be a culture change to move into an environment where management is willing to accept that long-range prediction dates cannot be made early with complete confidence.

4.3.3 Memorandum of Testing Agreement

Another area that is critical to comprehensive planning for software development is testing. Testing is possibly the most critical activity of the lifecycle of software development. In order to deliver a system that meets the functional business needs of the customer, extensive testing must be conducted. The testing verifies data security, role security, and acceptable functionality of the software. Sometimes testing is viewed as an annoyance, by both the development team and the customers, because it takes time away from already overloaded

schedules. However, thorough testing is imperative to the success of developed software. Only the customer can adequately test the software.

To meet the goal of accomplishing thorough and complete testing and to ensure an understanding and gain commitment for this critical activity, an agreement was made between the development team and the customer community. Both the project leader and the key customer signed this agreement. A copy of the agreement can be found in Appendix E.

Often the customer begins testing with sincerity to do a complete and comprehensive test, but distractions interfere, and testing is not adequate. The customer may then express confidence in the developer and want the software delivered anyway. Do not make this mistake. The project leader must ensure that adequate testing is conducted, regardless of its popularity. Errors are much more difficult to fix once the software is operational. The Testing Agreement Memorandum helps to reiterate the testing agreement and enforce compliance.

4.4 SOFTWARE PROJECT IMPROVEMENT IMPLEMENTATION FOR QUALITY ASSURANCE

The purpose of software quality assurance is to provide management with appropriate visibility into the process being used by the software development project and into the products being built. Software quality assurance involves reviewing and auditing the software products and activities to verify that they comply with the applicable procedures and standards. Software quality assurance also includes providing the software project leader and other appropriate managers with the results of these reviews and audits.

For quality assurance improvement, a goal was set to improve the review process. To meet this goal, this project initiated a series of technical reviews. Technical reviews are needed because, although developers are good at catching some of their own errors, many errors escape the originator more easily than they escape others. Three reviews discussed in this paper are:

- A Data Transition Review to validate the integrity of the code used in the transition
- A Data Design Review to validate the integrity of the data model
- A Customer Acceptance Review to validate the customer's satisfaction with the integrity of the transitioned data

The diagram in Figure 4-3 shows where each of these reviews is conducted in the development lifecycle. Additional information on transition was located in Section 3.

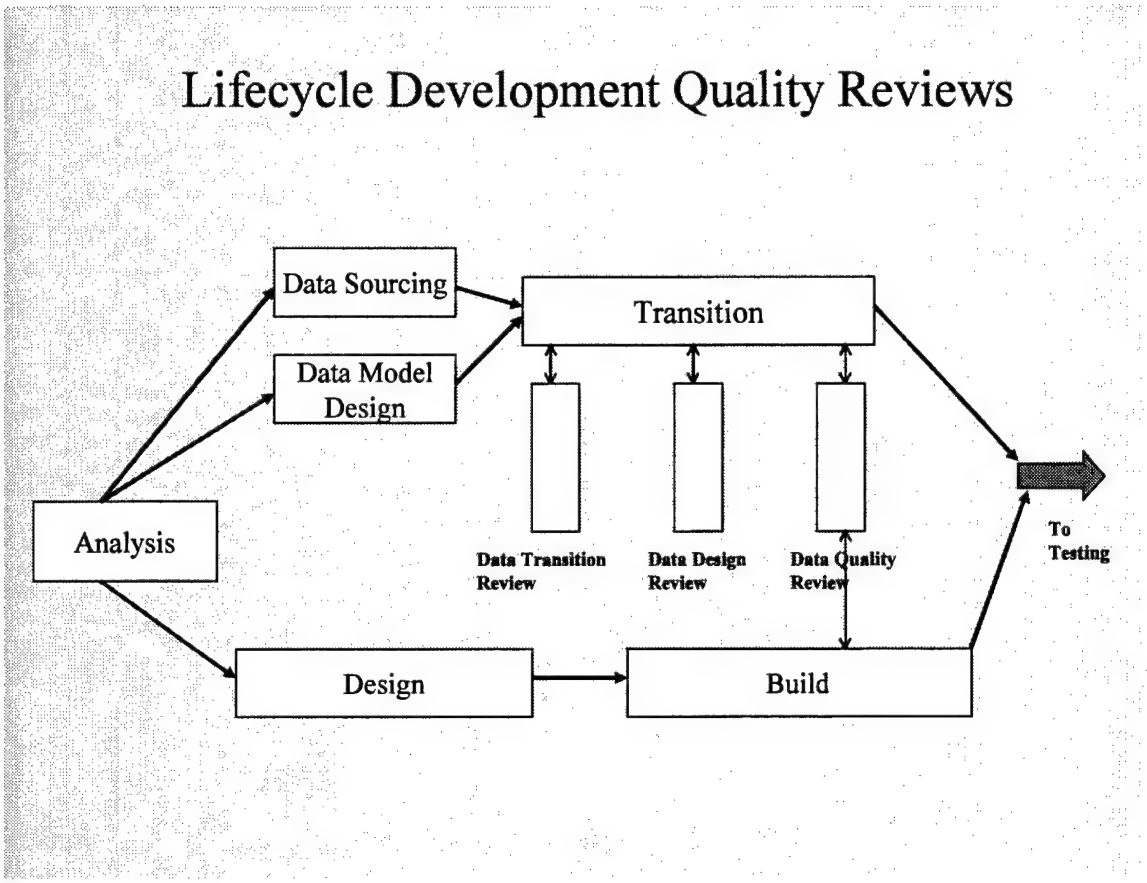

FIGURE 4-3. LIFECYCLE DEVELOPMENT QUALITY REVIEWS

The purpose of the reviews is to:

- Detect and remove defects early and efficiently
- Develop a better understanding of the defects that might be prevented
- Achieve more predictable quality than can be achieved without a review
- Provide management with reliable information indicating progress in the software development project by ensuring an acceptable level of usability in subsequent data transition activities

Each review committee was comprised of a leader, a recorder, and reviewers. Each participant had a defined set of responsibilities. Care was used in selecting review participants, especially the review leader. The participants were knowledgeable of the business and the data used in the business, and had, along with their manager's support, the necessary time to participate in the reviews.

4.4.1 Data Transition Review

Documented procedures in the form of checksheets and reports were developed for the data transition requirements review. The procedure checksheets and the technical review summary report (see Appendix F) ensure the following:

- All source data is mapped correctly to the data targets or accounted for via some document resolution

- The contents of the data sourcing information repository have been accurately translated into the rules database that contains input for the transition engine

The development team spent a day in preparation for the review. The morning was spent in gaining an understanding of how to use the materials. A detailed explanation of using reviews as measurement tools can be found in Reference 9. The afternoon session focused on conducting a dry run of the review process, prior to involvement with the customer. The development team suggested improvements both to the format of the document and the process. This meeting also served as a training session for the developers who would participate in the review. The changes were made and the review conducted successfully.

This paragraph describes the data transition review process. The reviewers reviewed all data transition requirements. Checklists were completed for each target table. Actions, issues, and related issues were documented. The reviewers examined the unmapped sources report to verify all unmapped source columns were intended to remain unmapped. For each table, the reviewers compared the Procedural Language/Structured Query Language (PL/SQL) transcripts to the contents of the data sourcing information repository. Results of the review were compiled, evaluated and presented to the appropriate persons. Initially, all six of the team members worked together. As the team developed expertise in the process, they were able to work in smaller groups, thus expediting the process.

As a result of the reviews, a more accurate transition of data was accomplished. By the time the final transition was made, the data was transitioned cleanly and accurately requiring only minor tweaking after the delivery of the software.

4.4.2 Data Design Review

The procedures developed from the Data Transition Review were modified to support the Data Design Review. This is one of the benefits of a repeatable process. The review process can be used, with modifications, for any review within this software development lifecycle methodology.

The objectives of the Data Design Review were to:

- Ensure all components of the data design specifications have been documented

- Examine all components of the data design specifications for completeness
- Examine all components of the data design specifications for conformance to standards
- Examine all components of the data design specifications for conformance to the rules of normalization
- Correct identified defects in the Data Design Review or document as issues those defects that cannot be quickly resolved
- Document all related issues

In Figure 4-4, the process developed for ensuring the quality of the database design is presented. The process is repeatable and is being used for other design efforts.

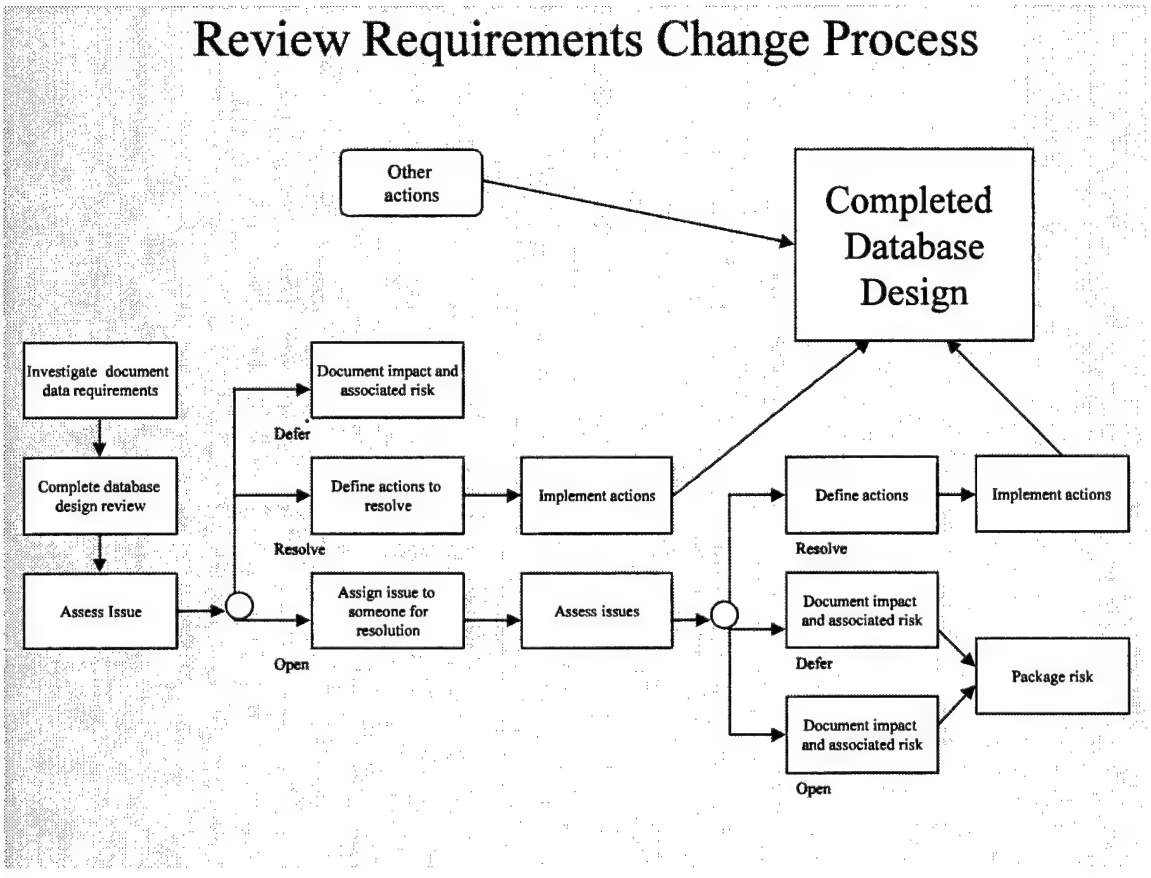

FIGURE 4-4. REVIEW REQUIREMENT CHANGE PROCESS EXAMPLE

This paragraph describes the Data Design Review process. The reviewers reviewed all static tables. Actions and related issues were documented. The following checks were conducted:

- Mandatory table properties
- Table/Key checks
- General column normalization and domain checks
- Mandatory column checks
- Detailed column checks

A sample of the checksheets is included in Appendix G. Results of the review were compiled, evaluated, and presented to the appropriate persons.

The results of the review indicated that many columns failed the checklist criteria. Evaluation of the errors indicated no systematic analysis had previously been conducted, allowing many errors to slip through analysis into transition. The team did not originally have an understanding of the review process goals and as a result did not incorporate the review process goals into previous work. Fortunately, because of the review, these errors were found before the application was built on top of a poorly defined database. The necessary design changes could then be implemented early in development, prior to transition and any application design or coding.

As a result of the reviews, a more accurate database design was developed, along with a higher quality of transitioned data. Because of the improved coding processes, decreases in errors were noted in other data design activities for another area of the project. The speed of the reviews increased because of the team understanding and education in regard to the review process.

4.4.3 Customer Data Quality Review

Once the Data Design Reviews and Data Transition Reviews were conducted, the structure and the contents of the data design specifications were updated. The changes were applied to the database design, and a transition was run to populate the database with data. Reports were developed to meet customer requirements and tailored ad-hoc reports were run against the database. These reports were reviewed in a series of workshops with the customer to ensure that the quality of the transitioned data was acceptable.

The review process for data quality is detailed in Figure 4-5.

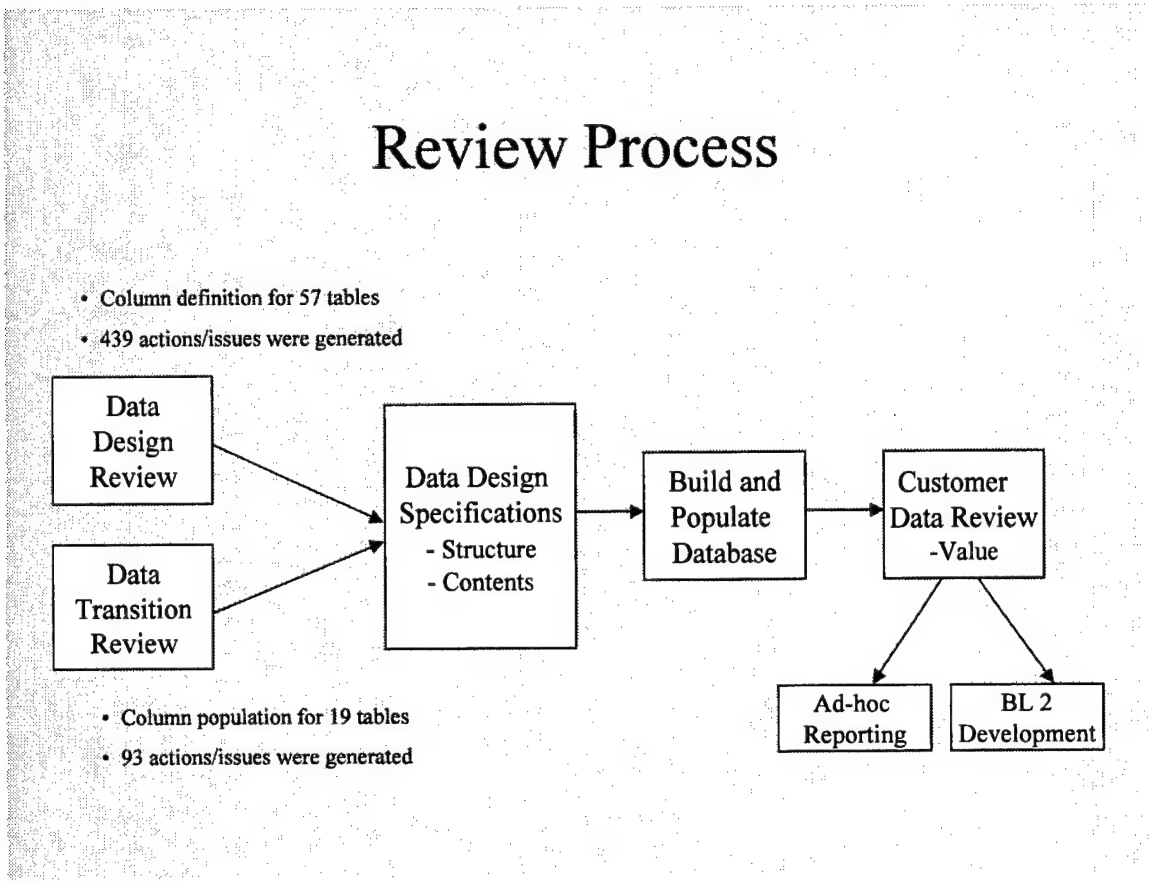

FIGURE 4-5. SAMPLE REVIEW PROCESS DIAGRAM

Generally, it takes several iterations and refinements of the design and transition of the data to gain customer acceptance. Final acceptance of data quality is formally documented during user acceptance testing.

The benefit from this review resulted in the customer actually seeing and using the data prior to delivery. Because of this data usage, errors that would have gone unnoticed until after the release were discovered and corrected.

4.4.4 Data and Database Design Review Lessons Learned

The following lessons were learned from conducting the reviews described above.

Transition:

- Failures were noted in the transition population scripts, the data sourcing information repository, or both. The reason for this was that the data sourcing information repository was developed from a customer perspective. As a result, the population of

the rules database did not support the technical level required, and thus, the code generated from the rules database was not explicit enough for code generation.

- The results of the reviews indicated the script programmers did not have adequate business knowledge to translate from the rules database.

- In the future, we should separate the data validation from transition development and have a data expert responsible for transition.

Reviews:

- Conduct analysis reviews prior to data design; conduct design reviews prior to transition review.

- Update and improve development standards (development standards needed to be rigorous, enforced, and continuously updated).

- Improve review preparation activities.

- Improve estimation techniques of time required for reviews.

- Require detailed and consistent documentation of design exceptions.

Data Validation:

- Improve the integrity checking and validation for transitioned data.

- Have transitioned data ready for testing of modules and reports.

- Develop formal procedures for validation and sign-off of data acceptance.

- Require more involvement by the customer community in data validation.

General:

- Working in teams (1) allowed a redistribution of knowledge so each team member had a broader-based knowledge, which reduced bottlenecks for information resolution; and (2) fostered team interaction, and over time, members of team became more concerned with solving problems rather than how or by whom they were caused.

- Errors began to decrease because of the improved coding process.

- Quality of the transition improved because of more accurate database design.

- Review speed increased based on process improvement and team education.

4.4.5 Module Design Workshops

Traditional development efforts use a paper-document-based approach for collecting requirements. These efforts generally have long development lead times and a low degree of customer involvement. Often, the first time the customer sees the software is when testing begins. As a result, the software developed has a low acceptance rate. Costs are significant for rework, and there is a low chance of fully satisfying the customer.

To support the implementation of quality software, a goal was set to improve the accuracy and completeness of the specification requirements. To meet this goal, an approach of iterative software development using functional design workshops was implemented.

Iterative software development approaches are gaining popularity. In this approach, high-level requirements are identified and then modeled, reviewed with the customer, refined and remodeled. This allows the developer and customer the flexibility of working with requirements that change between analysis and the final product. New generation of development tools support the evolutionary design. The customer sees what he is going to get and provides necessary feedback as software is developed.

There are risks associated with iterative development, but they can be managed. Scope creep is common. As customers "see" and "understand" the functionality of the software, they have new ideas and may wish to change the requirements. This could result in an endless change cycle of development. Another problem is premature requests for the product. The customer sees the software working and does not understand the immaturity of the software and indicates willingness to assume responsibility for early delivery. Nothing could be more disastrous. Finally, the developer has to know when to say when. As the software evolves, the developer sees better ways of implementation, but again, this process could revert to the endless change cycle of development.

Specific steps were taken to manage the scope creep. The customer was educated on the impact of requesting changes and became proactive in understanding what was critical to the initial release and what could be delayed. The SOW was often consulted if additional requirements were requested. Although some changes were added, the SOW set a baseline that was used for concurrence of the scope of the effort. The customer liaison worked closely with the customer community to help minimize the additional requests that were presented to the developers. The developers and customer community actively worked together as a team to minimize changes to the design.

Critical success factors for iterative development include:

- Maximized task congruence
- Structured walkthroughs
- Highly skilled development team

- Flexible development tools
- Early data transition
- Continuous customer involvement

The objectives for the design process include establishing a consistent, repeatable process for defining system specifications that satisfy the requirements of the business process. This is in keeping with CMM goals. Developers need to be enabled to increase capability to provide customers with products that have lasting value. Construction of a business process model and conducting design workshops are tools to be used to establish this process. The process is depicted in Figure 4-6.

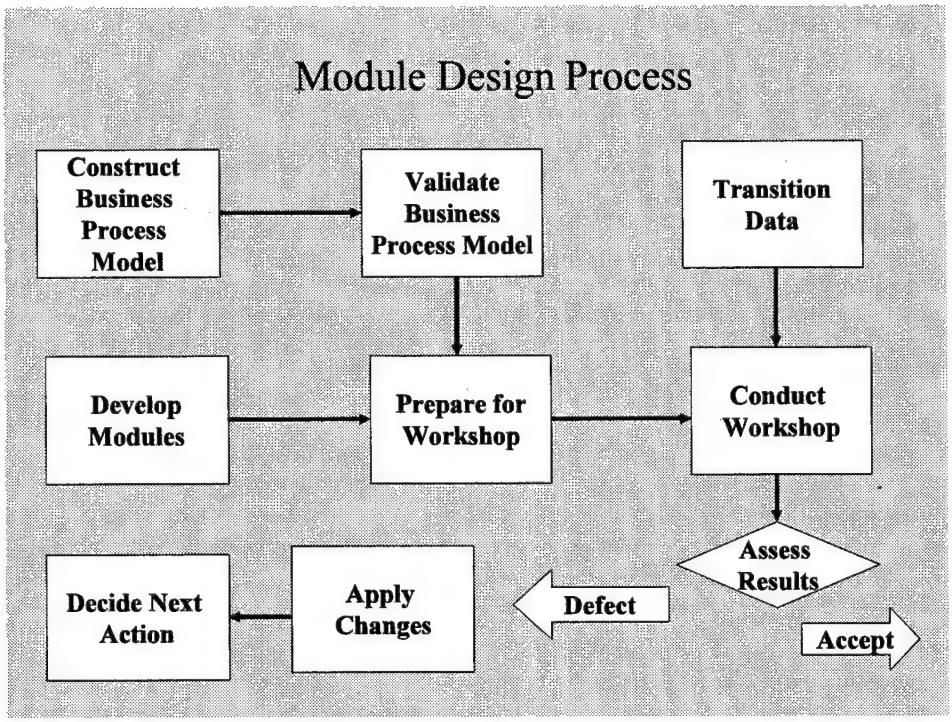

FIGURE 4-6. MODULE DESIGN PROCESS

Business process models help the developer and customer understand and communicate the vision, aims, objectives, needs, context, and financial success of a business. The models enable a business to achieve success by providing more value to the customer. The models help the developer master the details of the operation of a business so carefully selected parts of the business can be automated. To improve the quality and accuracy of the system requirements, a review of the business process for each functional area was added to the module design process. During the design workshop, the business model was validated and inspected for accuracy, completeness, and consistency for the intended viewpoints. Additionally, the inspection ensured

the developer understood the process adequately enough to proceed with effective automation of the process.

The purpose for module design workshops includes:

- Provide a structured environment for engaging the customer.
- Establish the business process model as the context for presenting the modules.
- Apply predefined criteria for finding defects in the modules specifications.
- Complete the module specifications.
- Make the outcome of the workshops visible.

To conduct successful workshops, the following guidelines were followed:

- Prototype of modules to review must be small enough to walk through in two hours.
- No single item is to be discussed for more than a few minutes (generally five minutes was the cutoff time).
- The product to be reviewed should be available for at least 24 hours prior to the walkthrough.
- Discussion items are to be recorded and distributed to the team after the walkthrough.
- Disposition for discussion items must be reached during the walkthrough.
- Attendees must be critical to the walkthrough and limited in number.
- Observers must limit comments to breaks and after the walkthrough.

All participants of the workshop had defined roles. These roles were as follows:

- Facilitator, who enforces guidelines and presents the business process
- Scribe, who performs item tracking and disposition recording
- Presenter, who presents the product
- Reviewers, who are the functional experts, and review the product to determine the accuracy of the product

- Observers, who attend by invitation only, but do not participate in the walkthrough

The activities conducted to prepare for the workshop were the following:

- Determine the business process area that will be reviewed
- Prepare screen flows and menus
- Generate first-cut modules
- Develop walkthrough scripts
- Set up workshop facilities
- Prepare for workshop using established procedures and checklists (see Appendix H)
- Prepare notification letters and packages for the attendees

A diagram of the steps conducted in a workshop is presented in Figure 4-7.

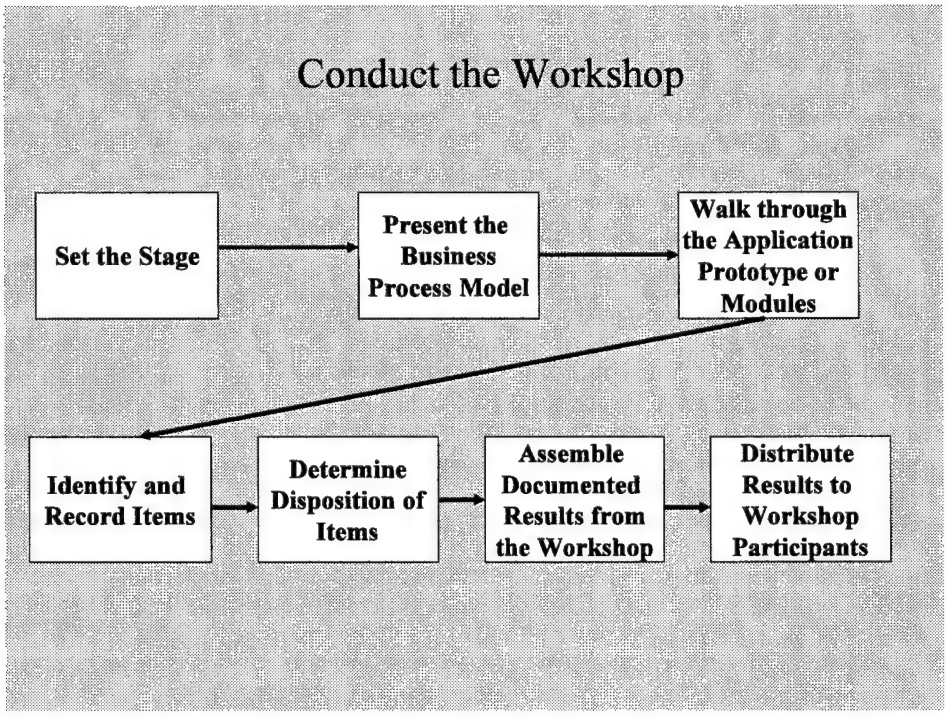

FIGURE 4-7. WORKSHOP ACTIVITIES

The following steps were conducted in the workshop:

- The facilitator sets the stage for the workshop by stating the purpose of the workshop, briefly describing the workshop procedures and guidelines, introducing the team members, identifying the customers, and reviewing any outstanding issues.
- The business process model (workflow) is presented to establish the context for presenting the modules.
- During the walkthrough, the high-level screen flow is reviewed, the standard look and feel is reviewed, the help screen is used as an aid to explain the modules, a walkthrough of data entry is conducted, and a walkthrough of reports is conducted.
- Module deficiencies are identified and discussion items are recorded.
- The disposition of items is determined.
- The outcome of the workshop is assessed.
- Visible consensus on the outcome of the workshop is noted.
- The results from the workshop are documented.
- The results from the workshop are distributed to the participants.

During the workshop, the facilitator reminds and encourages the participants to be selective when asking for changes. The "MosCoW" technique is recommended. Classify the changes as "Must have, Could use, Would like to have someday." To help the customers understand the impact of the change, the developer provides quick high-level assessments. As workshops progress, the customer becomes more proficient in knowing what they *must* have versus what they *want* to have. However, in the beginning, customers have a tendency to ask for everything. If change is not controlled, the software will never reach completion.

4.4.6 Testing

Software quality assurance includes developing standard and consistent processes for delivery of a quality product on a repeatable basis to the customer. A goal of this effort was to develop testing processes that could be used repeatedly to deliver high quality software.

Without adequate testing, no software application should ever be deployed. The risk is too great. Many projects fail in the area of testing. Often, there is no point of return when a new system is deployed prematurely. Customers cannot do their work and ultimately a business can be damaged permanently.

One such company that experienced this nightmare is Oxford Health Plans. From a recent *Business Week* (Reference 10): "Turnover among Oxford's programmers was unusually high, so the development effort lacked continuity. Much of the resulting work, say those who left, was poorly done. Moreover, they say efforts to test the new programs were minimal." Oxford was months late in getting bills to customers, customers canceled policies, stock plummeted, and at the time of this writing, the cost to the company was over $170 million dollars. Testing is critical to the success of an application. No parallel operation of the old system and the new system is needed, if testing is comprehensive and thorough and supported by adequate customer involvement.

Several levels of testing were an integral part of the development methodology on this project. Development of testing procedures for GUI software was quite a challenge. Because of the point-and-click capabilities and an endless number of ways to navigate through the software, no complete setup of testing procedures could ever be defined. After an unsuccessful attempt to develop traditional test procedures, the process was changed. Procedures were developed that were functional in nature. Instructions on how to navigate were seldom defined. Using a large group of testers, navigating in various methods to find defects worked quite well in testing the software.

Because of the complexity of the software, a team of knowledgeable business customers and developers of the software was required to produce the test procedures. Covering the entire testing process is beyond the scope of this document; however, an example of the test procedures developed is included as Appendix I.

Testing is an iterative process. Adequate time needs to be included in the schedule for rework and regression testing. One of the contributions this project leader made was to personally pretest every software component prior to customer testing. The project leader does not generally have the in-depth knowledge of either the customer or the developer and will find many opportunities for correction that might be overlooked. During the last few months of development, a full-time test person with extensive business knowledge joined the development team. This person spent an incredible number of hours testing the software in order to produce the test procedures. This tester found many, many errors. The lesson here is that there cannot be too much testing.

The diagram in Figure 4-8 depicts the different types of tests and where they are conducted in the development lifecycle.

4.4.7 Development and Testing Lessons Learned

Some of the most significant lessons learned are documented below.

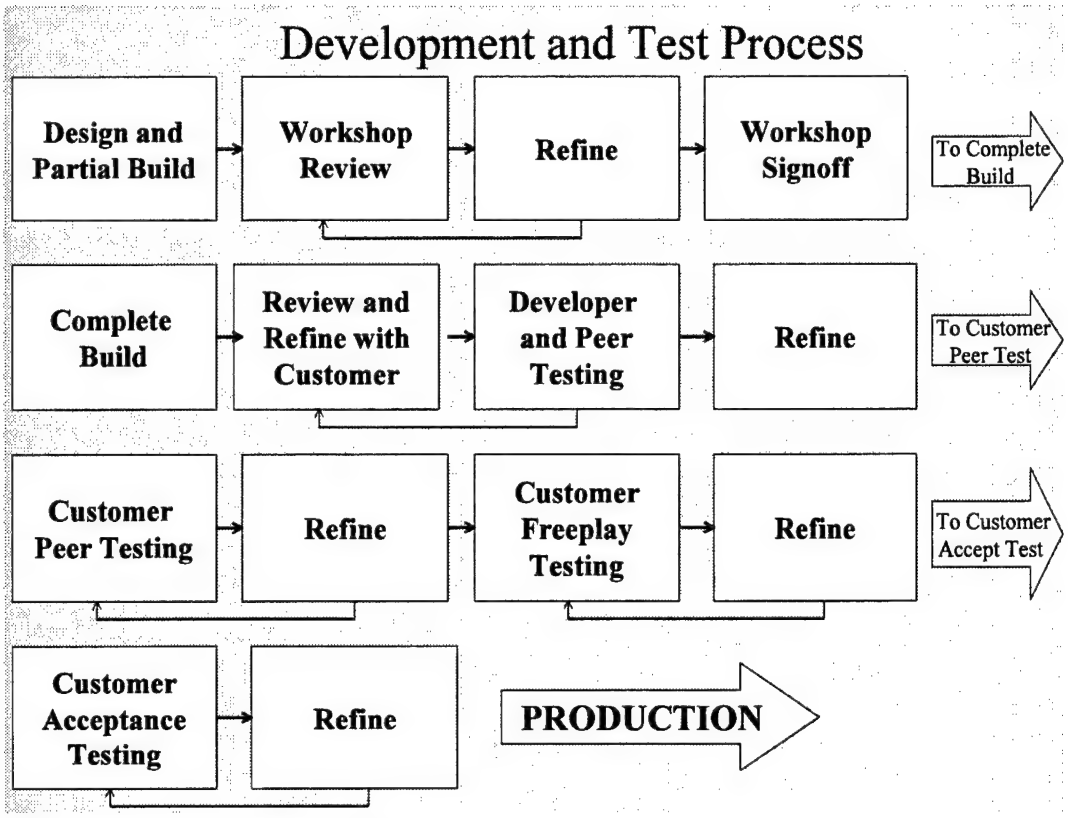

FIGURE 4-8. DEVELOPMENT AND TESTING PROCESS

Development:

- Develop all portions of the application in tandem. During the project, the common report interface was left until near the end to complete and caused rework of the reports. Ensure that the reports are ready when the modules are ready.

- Implement more structured peer reviews of design implementation plans and of generated code.

- Use the generation tools more effectively and limit tailored solutions. All tables, view, triggers, etc., should be generated from the repository.

- Develop common code and routines before the development of modules. Do not put code behind the screens to show functionality at a workshop. The code is hard to remove later and may even fool the developer into thinking the code is complete.

- Get the software into the hands of the customer as soon and as often as possible to improve decisions that impact design.

- Find more effective means of explaining to the customer the impact of making or not making design changes. Customers agreed certain workarounds were acceptable, but really did not understand the impact.

- Do not use the workshops to create a design prototype. Workshops are for requirements validation and screen and report layout confirmation, and are not intended to show the final product.

Development Configuration Management:

- Improve the development of the configuration management process and documented work procedures and the enforcement of standards.

- Document all requirements, business processes, and design decisions fully in the CASE repository.

- Apply configuration management to all versions of the application, including the development database, the testing database, and the production database. Valuable time was lost during this project because these three were not always in sync. Never let the repository get out of sync from the design during development. Apply configuration management to all database objects and transition scripts. Development needs to be baselined at all stages of testing.

Testing:

- Conduct more testing at the customer's workstation.

- Increase the rigor of testing by using a defined data set that allows results to be quantitatively inspected, rather than using all data transitioned for testing.

- Develop routines to test all database programming to ensure that no code goes untested.

- Improve the rigor of regression testing.

4.5 QUALITY: A BY-PRODUCT OF CUSTOMER INVOLVEMENT

The success of a software project has a direct correlation with customer involvement. The days of developers going off into a "black hole" with a set of customer requirements, developing the system, and delivering it to the customer with minimal or no customer involvement are over. In order to meet the customer's requirements without constant rework, the customer must be involved in the entire lifecycle of development. This project did just that. It is an accepted philosophy—the earlier the error is discovered in the lifecycle of development, the

less expensive it is to correct. Figure 4-9, obtained from Oracle's *CASE*METHOD*, provides a magnitude representation of the cost of error detection.

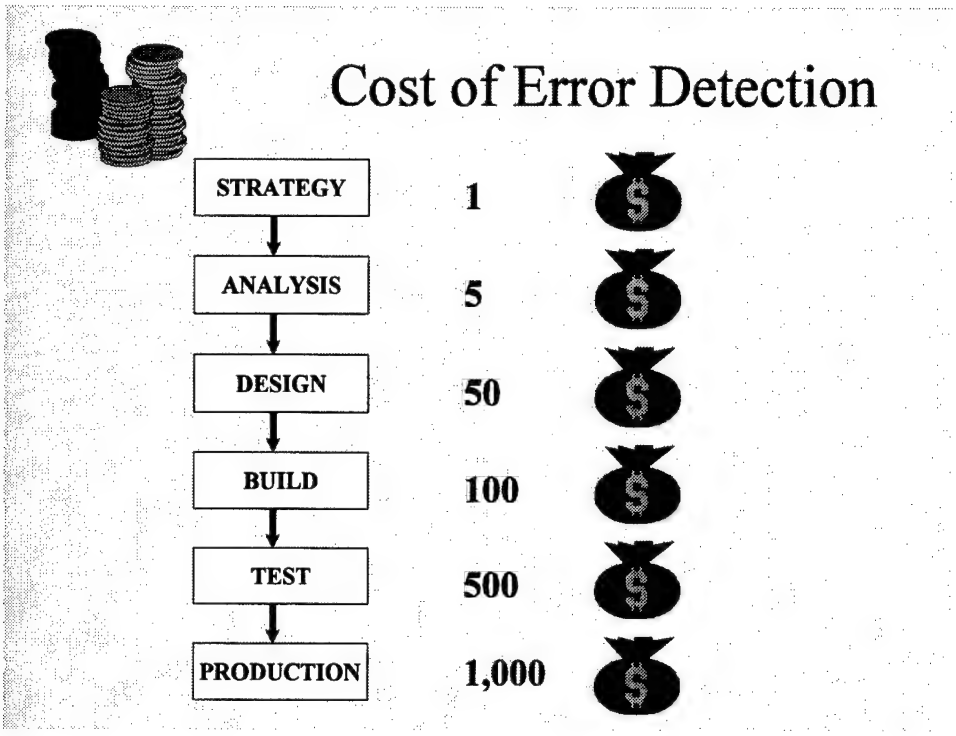

FIGURE 4-9. MAGNITUDE OF ERROR-DETECTION COST DURING DEVELOPMENT LIFECYCLE

As important as customer involvement is to obtaining a quality product, effective management of the customer is critical to success. The more customers involved, the more disruptive the development environment can become. If the developers are interrupted so much they cannot focus on the development, frustration will occur and schedule slippages will result. The previous topics in this section dealt with structured ways to work with customers; however, informal customer involvement is always present. In this project, the development lab and many of the customers were located in the same facility. Customers knew where the lab was and often stopped to have conversation with the developers. It is important to foster good communication and rapport between the customers and the project team, but the developers must be allowed to do their work.

Weinburg, in Reference 11, provided a suggestion for a solution to the constant interruption. He suggested hanging a sign on the door that said "Is this interruption Worth $100?" This dollar figure was based on a calculation of the amount of time and the cost of the developer. The figure could be higher or lower but the principle is the same—interruptions cost.

Two approaches were used to reduce customer interruptions. At every meeting or review, customers were reminded by the project leader that the developers were interested in feedback and customers were encouraged to call the developer and set up a time to meet. They were also

reminded that too many unplanned interruptions could actually delay the delivery of the software. On certain occasions, the project leader simply shielded the team by conducting important working sessions and meetings away from customers, and, if necessary, posting a sign on the lab notifying potential interrupters that the developers were not available. Generally, the customer community was supportive in understanding the development environment and following the established customer involvement suggestions.

SECTION 5

PROJECT MANAGEMENT

Project management has and continues to evolve. Each project is unique, and thus the management of the project is unique. Many authorities and references exist to assist the project manager, but it takes a skilled and determined project leader to develop the successful formula for management of a project. This section provides background, insight, and suggestions for accomplishing effective project management. According to Joiner in Reference 12, there are four generations of management.

First Generation—Management by Doing:

This is the first, simplest, most primitive form when you just do it yourself. We still use it and it is effective, but its capacity is limited.

Second Generation—Management by Directing:

People found they could expand their capacity by telling others exactly what to do and how to do it. This approach allows an expert to conserve time by getting others to do the work in compliance with strict standards.

Third Generation—Management by Results:

When people get frustrated by being told how to do every detail of their jobs, the manager can tell the person what to do and leave it up to the person to figure out how. Reward or punishment is based on how well the job was done.

Fourth Generation—Management by Quality, Scientific Approach, and the Team as One:

Fourth Generation Management compensated for the limitations in each management generation and became the champion of customer needs and driver of real improvement. These managers work together with other employees to help develop better and better methods to get better and better results.

To facilitate an understanding of Fourth Generation Management, Joiner has developed the Joiner Triangle that is shown in Figure 5-1.

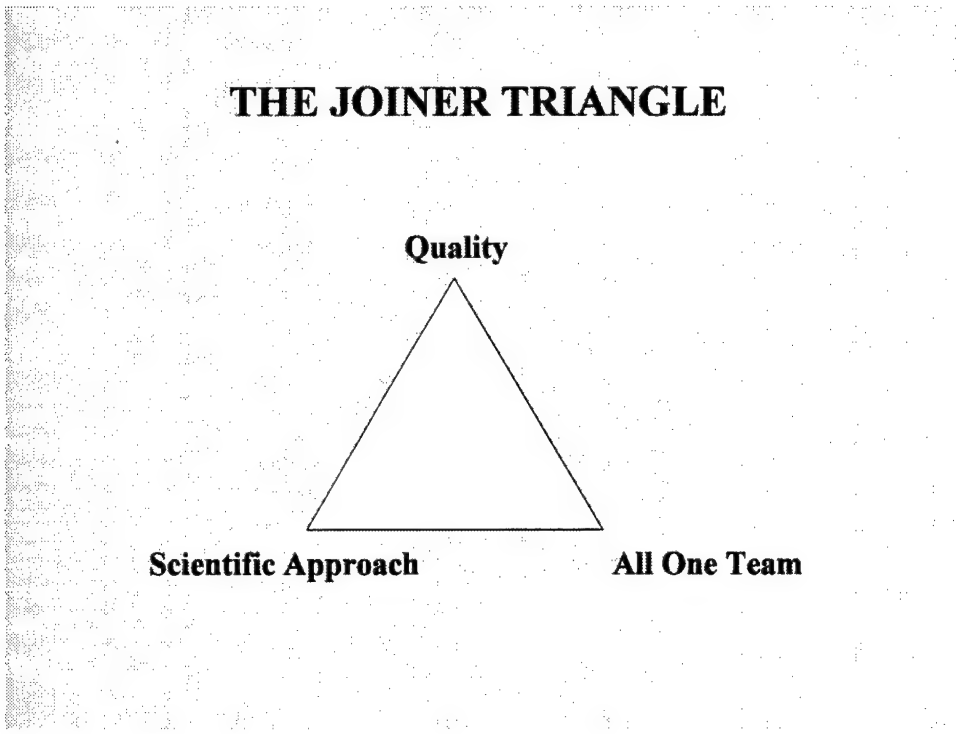

FIGURE 5-1. THE JOINER TRIANGLE

The corners of the triangle are:

- Quality: Understanding that quality is defined by the customer; developing an obsession for delighting customers—not just being satisfied with merely getting rid of what annoys them but going beyond to understand their current and future needs deeply, to surprise them with products and services they didn't even know were possible. This understanding is no longer the domain of special groups within the organization; rather, it is shared with and further developed by every employee.

- Scientific Approach: Learning to manage the organization as a system, developing process thinking, basing decisions on data, and understanding variation.

- All One Team: Believing in people; treating everyone in the organization with dignity, trust and respect; working towards win-win instead of win-lose for stakeholders (customers, employees, shareholders, suppliers, the communities in which we live).

By finding out what is important to the customers and spending resources wisely, everyone, according to Joiner, comes out a winner. These three elements seem obvious at first, but these elements do not come naturally to the project leader. Quality and productivity are two sides of the same coin, not opposing views. Viewing our organizations as systems helps to achieve a customer focus and reduce waste and inefficiency. To improve, rapid learning is

required, and then rapid application of the learning to achieve improvement. This project leader applied these principles to this software development effort.

Quality requires the necessary time and resources. If time is shortened, quality is impacted. If resources are lost, time is impacted. Any change to the needed resources will affect quality. The project leader must exhaustively fight for realistic schedules and necessary resources.

Although not every learning opportunity can result in a successful improvement, these opportunities must continually be identified, analyzed, and implemented when appropriate. Some of the management philosophy for development is identified in the following section to facilitate other project leaders gaining an awareness of techniques of self-directed and empowered management.

5.1 TEAM COMPOSITION

In order to motivate personnel, a talented, dynamic, and cohesive team is required. Often the project leader is not given the authority to select team members. Not empowering the project leader to build an effective team composition often spells failure from the beginning. The project leader must consider more than qualifications. Other attributes such as personality, ability to participate on a team, and working well with others could be critical to the success of the project. Team members must build a respect for each other, and to foster this respect, team members must have the ability to successfully participate and contribute to the project's success.

The project manager should not attempt so much to motivate the team members, but rather carefully select team members who will be motivated and work creatively for a project. Meridith and Mantel provide guidelines for acquiring and motivating personnel in Reference 13. From their guidelines, the project leader selected the criteria she felt most critical to the success of the project. Listed below is the project leader's selection from their recommendations of the most important criteria considered that were used to find effective team members.

- High-Quality Technical Skills: Team members must be able to solve most of the technical problems of a project without recourse to outside assistance.

- Strong Problem Orientation: Greater chances for success exist if the team members are problem-oriented rather than discipline-oriented. Problem-oriented people tend to learn and adopt whatever problem-solving techniques appear helpful, but discipline-oriented individuals tend to view the problem through the eyes of their discipline, ignoring aspects of the problem that do not lie in the narrow confines of their educational expertise.

- Strong Goal Orientation: Projects do not provide a comfortable work environment for individuals whose focus is on activity rather than on results. Workflow is rarely even, and for the professionals, a 60-hour week is common, as are periods when there seems little to do. "Clock watchers" will not be successful team members.

- High Self-Esteem: Projects can rapidly get into trouble if team members hide their failures, or even a significant risk of failure from the project manager. Individuals on the team should have sufficient self-esteem that they are not threatened by acknowledgement of their own errors, or by pointing out possible problems caused by the work of others. Egos must be strong enough that all can freely share credit and blame.

Selecting the project manager is one of the most important decisions to be made about a project. The following list of attributes, compiled by the author from various sources and experience, can be used when selecting project managers.

- Strong technical background
- Mature individual
- Willingness to take risk
- Willingness to defend the team members
- On good terms with senior management
- Person who keeps the team energized and happy
- Excellent people skills
- High self-esteem
- Self-starter
- Ability to sense and remove interpersonal conflict
- Ability to keep the project on schedule
- Willingness to do whatever it takes to get the resources or get the job done
- Ability to handle stress
- Charisma
- Leadership abilities

Quite a bit could be written about each of the above characteristics, but most are self-explanatory. However, more needs to be stated about leadership. A manager is not necessarily a leader. Giving a person the title of project leader will not assure the project's success.

Characteristics above, such as technical credibility certainly help, but leadership is more than authority. Leadership cannot be taught. It comes from within the leader and the power granted to the leader. Real power is referent power that the project leader earns from the team members.

Training is imperative for the success of the project leader. Wesner, Hiatt, and Trimble identify, in Reference 14, the training that is recommended, which includes the following:

- Leadership skills
- Communication and facilitation skills
- Training skills
- Consulting skills
- Problem-solving skills
- Feedback skills

This project leader would also add technical training to the list of required characteristics. To effectively manage and accurately predict schedules, the project leader needs at least a working knowledge of how the software is developed, the tools that are used, and the complexity of the tasking.

In this project, the team was a mix of many persons from six different contractors, under two contracts, and government employees. What is important to note is that the cohesiveness of the team is driven by selecting the right persons for the project, not selecting people because they all work for the same person or company.

5.2 TEAM DYNAMICS

At the time the project leader joined the effort, the members were experts in character-based Oracle development; however, the members had no experience in GUI client-server development. The project leader hired contractors to provide that experience. A System Engineer with substantial GUI background joined the team and took a leadership role. This person shared her GUI knowledge, and the team members shared their Oracle and business knowledge. The project leader also hired a consultant to work with the effort for a six-month period to establish process improvement initiatives and train the team in JAD workshop techniques. A client-server expert was also hired to develop the approach and mechanism for deployment. By hiring experts, where the original team lacked expertise, minimal training was required. With the Oracle background covered by the original team and the GUI background covered by the additional system engineer, only Oracle GUI training was needed.

Positive team dynamics began shortly after the project leader hired the necessary expertise and set the development ground rules for the team. Initially, there was some skepticism aimed at the project leader because of the rigor and structure that was brought to the effort. Much later in the effort, several team members said they were doubtful the effort could be turned about, but to their surprise, it did turn around quickly. There was doubt about the implementation of repeatable processes and quality improvement initiatives. Once the team members saw positive results, they became supportive to the effort and the team was brought together.

In a large project, it is effective to establish the roles needed to support the effort. Once the roles are determined, the appropriate persons can be assigned as primary or backup for each role. A backup should be available for every role. Only 12 people were full-time, permanent team members, but creative assignment covered a wide area of responsibility. Table 5-1 provides a list of the roles that were covered by these 12 team members. (Note that the Customer Liaison is considered a team member.)

TABLE 5-1. TEAM ROLES

ROLE	**Primary**	**Backup**
Project Leader	Project Leader	
Analysis Business Expert	4 team members	
Developer Business Expert	4 team members	
Lead Software Engineer / Repository Manager	1 team member	1 team member
Module Screen Developer	4 team members	
Module Report Developer	1 team member	1 team member
Common Report Interface Developer	1 team member	2 team members
Version Control Manager	1 team member	1 team member
Adhoc Tool Report Developer	1 team member	1 team member
Workshop Organization/Facilitation	Project Leader	
Workshop Preparation	4 team members	
Database Administrator	1 team member	1 non-team-member
Client Server Architect	1 team member	
Transition Engine Developer	1 team member	1 team member
Pre-customer testing	Project Leader 1 team member	
Test Coordinator	Project Leader	
Customer Training	2 team members	2 team members
Preload, Translate Developer	1 team member	1 team member
Transition from old to new database	3 team members	
Old/New system synchronization	Support Developer 2	Support Developer 1
Application Cutover	Project Leader 4 team members 1 non-team-member	
Defect Tracking	Project Leader	1 team member
Customer Liaison	Customer Liaison	Project Leader

Early in the development effort, the project leader initiated activity to consolidate development into a central lab. Prior to the consolidation, many developers worked off-site at their company location. Much interaction and sharing of information was lost. After the lab was established, each developer had a workstation in the lab and was expected to spend a significant amount of time in the lab developing. The side-by-side interaction of the developers in the team lab encouraged team cohesion and sharing of information. During the last six months of the effort, the project leader also spent the majority of time in the lab. The project leader was able to observe team morale and to learn about concerns that might have otherwise gone unnoticed for longer periods of time.

Although budget constraints were always with the team, the lab had white boards and a team conference table. This provided the developers with the ability to communicate to others visually and collectively. Developers overheard conversations and provided suggestions and information. When one developer had a problem, another would come to his or her aid. Developers began to go to lunch together, exercise together, and share with each other. This interaction strengthened the team bond.

Having the lab near the project leader's office provided the opportunity to frequently, many times a day, enter the lab and interact with the developers. During the crunch times and high stress time, the project leader spent a minimum of several hours a day in the lab.

The project leader must be sensitive to the needs of the developers and the impact of working in the lab. Some team members needed quiet time and were encouraged to spend some time outside of the lab. Even though working in the lab at times is noisy and not appealing to everyone all the time, it is this project leader's opinion that the team lab environment was a major contributor to the success of the development effort.

5.3 PRACTICAL TECHNIQUES THAT WORK

W. Edwards Deming is a prominent leader in recognizing the importance of implementing quality in product development. From his work, many concepts have emerged in moving from an autocratic to democratic to self-directed management. Moving towards helping people to optimize the system so that everyone will gain takes years of work and commitment. This project manager attempted to apply many of the principles of Deming's new role of a manager described in Reference 15 in the development effort. The principles adopted are listed below.

- A manager helps people to see themselves as components in a system, to work in cooperation with preceding stages and with following stages towards optimization on the efforts of all stages toward achievement of the aim.

- A manager of people understands that people are different from each other. He tries to create for everybody interest and challenge, and joy in work. He tries to optimize the family of background, education, skills, hopes, and abilities of everyone. This is

not ranking of people. It is, instead, recognition of differences between people, and an attempt to put everybody in position for development.

- A manager is a coach and counselor, not a judge.

- A manager has three sources of power: (1) authority of office, (2) knowledge, and (3) personality and persuasive power. A successful manager of people develops (2) and (3); he does not rely on (1). He has, nevertheless, an obligation to use (1), as this source of power enables him to change the process to bring improvement.

- A manager will study results with the aim to improve his performance as a manager of people.

- A manager creates trust. He creates an environment that encourages freedom and innovation.

- A manager does not expect perfection.

- A manager listens and learns without passing judgement on him that he listens to.

- A manager holds an informal, unhurried conversation with every one of his people at least once a year, not for judgment, but merely to listen. The purpose would be to develop an understanding of his people, their aims, hopes, and fears.

5.3.1 Effective Meetings

Meetings that are not effective adversely affect productivity and morale. This project leader has found the following meeting guidelines to be effective:

- Provide an agenda to each individual.

- Be enthusiastic.

- Start ON TIME.

- Do not talk DOWN.

- If lecturing, so state at the beginning.

- If conducting a meeting, be specific and then BE QUIET.

- Take good notes.

- Smile.

- Do not make promises; commit only to seeing what can be done.

- Thank everyone for his or her time.

Another technique to use to achieve effective meetings is facilitation. Early in this development project, the project leader hired consultants to facilitate important meetings and working sessions, especially those involving customers. Also, the project leader received formal training in facilitation. Once trained, and having observed the consultants' effective facilitation, the project leader assumed the facilitation responsibility. (Additional details on facilitation were provided in Section 4.4.)

5.3.2 Accurate Schedule Predictions

Many approaches can be used to support accurate schedule predictions; however, it is beyond the scope of this paper to discuss the various techniques. It is assumed that the project manager has knowledge of Program Evaluation and Review Technique (PERT), the Critical Path Method (CPM), Gantt charts, and other schedule management tools. One certainty is—Without the use of these tools, it is difficult to manage a schedule. The tools allow the project manager to try multiple what-if scenarios and understand the impact to the entire project for what may appear only a minor delay. Use of automated scheduling tools allows the project manager to easily make changes and provides supporting reports and graphs to follow trends and progress.

Software development has become increasingly difficult to manage over the last decade. One of the main reasons is that managers can no longer effectively do the same work as the developer. Software is becoming so complex that specialists are required for many areas, with no one person technically able to accomplish all tasks. For example, a software development effort might have an expert dedicated to each of the following efforts: CASE tools, client-server deployment mechanisms, database administration, transition engine development, etc.

This leaves the project manager in a less than desirable position. Because the project manager has little technical knowledge in these highly sophisticated areas, he or she is virtually dependent on the team members to project completion dates. This does not produce accurate results because traditionally, developers are quite optimistic in predictions, having little project management experience from which to generate the big picture when predicting schedules. In addition, many of the highly technical specialists are often novices in actually doing the work. They have the technical expertise, but not the background from previous development efforts, so they really don't know how long something will take because they have not done it before.

To compensate for the lack of ability to predict schedule, the following steps are recommended:

- Even though the manager may not be technically able to do the work, the manager can spend time with the experts to ascertain how complex the activities are and how long

the activities generally take. Schedule sessions with each technical expert and watch him or her do assigned tasks.

- The manager can encourage the developers to better support their managers and customers by providing better schedule estimates. Accurate schedule estimates have valuable marketing potential. Often the developer does not know that reliable information of this kind is necessary for managers to do their jobs effectively.

- When possible, the manager should track metrics on individual development activities and use the metrics to understand the larger picture. For example, a developer could produce software modules of different complexity and determine how long each of those efforts took. Modules could then be classified as to complexity. Metrics obtained from the trial or early development could be applied to the classified modules to produce an estimate of time required to produce all the modules. As development progresses, the metrics can be refined, and the schedule predictability grows in accuracy.

5.3.3 Incorporating Tried and Proven Techniques and Expert Knowledge

Throughout this paper, philosophies and direction from many leading experts in software development, quality, and project management have been mentioned. The works of these experts were paramount in developing effective management and implementation strategies. In addition, time, dedication, and commitment are necessary for a project leader to learn and apply these principles. However, it is significantly less time-consuming to determine and incorporate effective strategies initially, before forging ahead without planning and having to constantly figure out what went wrong and how to remedy the problems. The author highly recommends the project leader continually expand his or her management expertise by ongoing study of the writings of experts.

The development team used Reference 16, the Department of Defense (DOD) MIL-STD-498 Application and Reference Guidebook, to support the development effort. This document can be used to support any standardization of any software development effort. It is beyond the scope of this paper to provide the details of how this document was used throughout the lifecycle of development, but the author would be remiss not to recommend this document for its significant contribution in supporting the development effort. The following description of the document is provided[16]:

> MIL-STD-498 is a standard for the software development process. It is applicable throughout the system life cycle. It establishes uniform requirements for acquiring, developing, modifying, and documenting software. It defines standard terminology and establishes activities, tasks, and products for a software development or maintenance project. It can be applied to any type of software, including application software, operating system software, the software portion of firmware, reusable software, and software employed to develop deliverable software.

The activities in the standard form a comprehensive set, sufficient for a large, complex project, but scalable and adaptable to suit the needs of small ones. The standard is meant to be tailored as needed for each project by specifying in the contract which provisions of the standard apply.

The standard is intended to be responsive to the rapidly evolving software discipline. It is independent of any particular methodology, ensuring the acquirer's right to specify the product and the developer's right to be technologically creative and innovative. In particular:

- The activities and tasks in the standard tell what to do not how to do it. For example, the standard requires the developer to perform architectural design, but does not require use of the object-oriented design method.

- The standard does not specify or encourage any software life cycle model (Waterfall, Incremental, Evolutionary, Spiral, etc.). Instead, the standard provides the building blocks needed to carry out the life cycle model selected for a software project.

- The standard does not specify or depend on any design or programming language. It is meant to be applicable regardless of the language used.

- The standard provides flexibility regarding documentation.

- The standard does not emphasize any particular software quality factor, such as reliability, maintainability, or reusability. This choice is left to each project.

- The standard can be used within an organization or contractually between two parties.

- The standard provides a performance-based distinction between requirements for a product and design of a product. A requirement is a characteristic that a system or software item is required to possess to be acceptable to the acquirer, while design is a characteristic that the acquirer is willing to leave up to the developer. This emphasis on required performance can allow a developer to provide quality products at reduced costs when unique product solutions are not required.

This project leader encourages other software development project leaders to acquire a comprehensive on-site technical library. The ready availability of the writings of experts will provide the project leader with additional background on which to base planning and decisions.

The reference list in this document includes most of the books that belong to this project leader's library.

5.3.4 Make Work Pleasurable and Rewarding

Software development is stressful even without critical deadlines that are always hanging over the heads of the team members. One of the best therapies for dealing with a stressful environment is to make it fun. A good mix of people helps this to happen, but it is imperative the project leader enjoys the challenge and brings some lightness to the team. Occasionally, take team breaks. The project leader can set the stage by taking a relaxed posture in the development lab and engaging the team members in conversation unrelated to work. It does not have to be a long break, but the interest shown by the personal interaction of the team members is like a rest break. No time is lost, because the members have taken a mental rest break and can refocus.

The best situation is where all developers like what they are doing. This is not always possible. The project leader needs to have an awareness of each team member's personal attitude in order to provide opportunities to give the developer as much satisfaction as possible. Hopefully, the developer has chosen that particular line of work; if not, the project leader should assess if this is a satisfactory arrangement both for the team and the person.

The state of mind of the project leader will spill over onto the team members. The project leader should be the barrier from unpleasantness and, as much as possible, is upbeat, optimistic, and energetic. Unhappy project leaders produce unhappy teams. Studies show the best results occur when team members are happy doing their work.

5.4 WORKING WITH SUBORDINATES MORE TECHNICALLY COMPETENT THAN THE MANAGER

When team members are more technically competent than the manager, a difficult challenge is presented. The manager must win the technical team member's respect in areas other than technical expertise. Although the leader may not have the level of technical expertise, the leader possesses knowledge, personality, and persuasive power and these are used to gain respect.

Technically challenged employees are more likely to remain for a longer time with the organization, so ways of challenging these persons is imperative to longevity of involvement. As stated by Glinow in Reference 17,

> The organization must be willing to provide continued training so that the specialist's skills will not atrophy, or become obsolescent. Organizational incentives should be designed to utilize a key attribute of the professional orientation - expertise.... For example, the quality of facilities and services.... have been found to be very important to highly skilled professionals.

The creation, distribution, and dissemination of information regarding appropriate incentives are considered critical in controlling the performance of people in organizations. However, review studies have shown that the administration of traditional rewards such as promotion, pay increases, status symbols such as window offices, private parking spaces and keys to the executive washroom are less effective in controlling the performance of professional employees.

Controlling the performance of highly skilled technical and professional employees appears contingent on the degree to which organizations can mediate the tensions between the professional's values, attitudes and beliefs and the organization's bureaucratic authority and hierarchical control systems. The manner in which organizations develop and use their professional talent depends, in part, on the incentives designed to motivate and control a professional's performance with key characteristics of the professional's orientation. This matching focused on alleviating problems inherent in professional-organizational conflicts. The neglect of professional employees through inattention to the evaluation and control of system, resulting in either obsolescence, dissatisfaction, or turnover, is a risk no organization should willingly assume.

There are many techniques a project leader can use to challenge and retain the technical employee. Some of these are described in the following paragraphs. This portion of the document is intended to provide a comprehensive overview. Although the project leader could not employ all these techniques, the techniques that were incorporated in this effort are annotated at the end of the task as "applied to this development effort."

5.4.1 Expanded Tasking

Organizations should provide for the growth of technical skills over time, to avoid professional obsolescence and job dissatisfaction. Acquisition of additional technical skills allows an employee to continually attempt challenging work. Examples of how tasking can be expanded follow:

- Grant leaves of absence for engineers and other technical professionals to attend educational opportunities, such as seminars, education programs, and sabbaticals to permit and encourage the strengthening of technical skills and expertise.

- Encourage career changes and rotation of tasking for obsolescent professionals, or high technology employees whose continued expertise requires different and/or additional cross-training.

- Initiate a structure where each employee is a technical guru in one area and an experienced backup in one to two other areas allowing each employee to learn additional skills and broaden his or her expertise (applied to this development effort).

- Encourage affiliation with technical and professional organizations and meetings where the highly technical employee can share ideas and gain knowledge from others (applied to this development effort).

- Provide opportunities for on-going advanced technical training (applied to this development effort).

5.4.2 Empowerment

Wilson in Reference 18 states, "Today organizations must deal with a hierarchy that blurs the lines of authority...Organizations deal with this change in hierarchy by creating working groups, quality circles, management teams or departmental committees. These "self-directed" teams are used as a basic organizational building block. People are recruited for their teamwork potential, freed from traditional bureaucracy and paid for their performance..."

Wilson stated that there are three main reasons why these teams sometimes fail:

- Because upper and middle management misunderstand the concept of the self-managing team

- Because some members see the team as an opportunity to 'empire build'

- Because team members, supervisor-leaders, and managers are poorly trained

The project leader of a highly technical team must work to eliminate the reasons for failure by finding practical suggestions and guidelines. How well the project leader handles these challenges, determines the success of the project. One approach is through empowerment.

To "empower" means to enable, allow, or permit by encouraging team members to participate actively in the decision-making process. This allows team members to achieve recognition and involvement and promotes job satisfaction and morale. In order to empower a team, the project leader must first empower herself or himself. This means participating, not just leading. It means accepting ideas and opinions other than your own. It means listening and providing positive feedback. Team empowerment will not just occur; the project leader must provide the environment and foster growth. Basically, the project leader must be willing to share responsibility and decision-making.

To be an empowered leader, you must demonstrate the confidence that you can take responsibility for the success of the team. Your personal feelings of self-worth are expressed through positive body language and statements. You must be committed and believe in yourself, the team, and the effort to be accomplished. This project leader attempted to practice the following philosophy so that the team could follow her example:

- Be proactive and encourage participative team members.

- Create opportunities for team members to express their thoughts, opinions, and direction.

- Actively listen and give feedback to other team members.

- Encourage the team to take responsibility for reviewing and evaluating progress.

Team members are often reluctant to openly communicate because they see the project leader in a power position. The project leader does not have any "real" authority. He cannot make things happen. The project leader must gain personal power conferred by the team by demonstrating skills, abilities, experience, support, and action. The project leader contributes to team empowerment by using "people skills." These skills can be taught, but will be ineffective if the project leader is not willing to give up an autocratic style of management. The project leader must learn to be open, not emotional, and not self-seeking for personal gratification. This is one of those situations where the leader must "walk the talk."

To achieve success in this area, this project leader encouraged shared responsibility by using the following steps:

- Setting up assignments that required the input and cooperation of several members

- Sharing as much information as possible

- Showing the team members how their particular task played an important part in the overall project

- Allowing team members to share their skills and knowledge with each other by encouraging "cross training within the team"

- Listening, and then sharing team disappointment, or celebrating team successes

5.4.3 Self-Inspection as a Motivator

J. M. Juran has had a monumental influence and profound effect on the field of quality management for more than 70 years. Most of his principles have been embraced by other quality authoritarians and documented in this paper. One technique this project manager borrowed from Juran, in Reference 19, is adopted from the self-inspection principle. This project leader provided very little direct guidance to team members. When a team member joined the team, the only direction given was a request for prompt attendance at meetings, and the team member was to stand by the quality of his or her work. The project leader explained to the team member that he or she was empowered to do the work as he or she determined effective, and as a result, the project leader expected quality software. If the team member said the software worked, then the project leader accepted the software with full assurance it was correct. The team member was responsible for determining whether the product conformed to quality goals.

Given this responsibility and challenge, the highest quality of software was produced, and the project leader could be certain of its performance. The only time the software was not of acceptable quality was a misinterpretation of requirements, and this was easily recognized by continuous customer involvement.

5.4.4 Job Time Redesign Approaches

In addition to job enrichment, other work design techniques are also growing in popularity. Three to consider are flextime, modified workweek, and working from home.

Flextime allows the employee some discretion in arranging work hours. In this project the project leader requested team members to be available between the hours of 9 a.m. and 12 p.m. and 1 p.m. and 3 p.m. Vecchico, in Reference 20, states that research on flextime has yielded generally positive results. Absenteeism and turnover are lower, while performance and productivity increase.

Modified workweek plans provide alternatives for the prevalent 8-hour day, 5-day week. Most common are the 4-day, 10-hours-per-day workweek or 4-day, 9-hours-per-day workweek with half a day off each week. In this project, this approach did not apply, as many team members willingly worked 5-days, 10-hours-per-day on their own initiative. However, no pressure was applied for employees to work long hours.

Working at home has merits, but is difficult to work around when team interaction and working with the customer is important. This project leader allowed some flexibility in this area. After one team member had a baby, she was allowed the opportunity to work at home one day a week for six months to ease the transition back to working on-site full time. Team members understand which days are most critical for being at the work place. Team members who are contractors are allowed to work at home occasionally. For example, if a team member needs to be home because of a repairman or a sick child and does not wish to take a day off, the project leader allowed them to work at home. It is the team member's responsibility to have the right equipment at home to be productive.

There is always the possibility of setting precedence, so each situation must be individually evaluated. In this case, giving the team members the ability to work flextime and at home, when needed, increased productivity and, certainly, improved morale was a result.

Job time redesign approaches do not redesign the actual tasks, but they do allow the employee a means of balancing work life and home life.

5.5 BUILDING TEAM MOTIVATION

One of the challenges about being a motivational project leader is remembering, reinforcing, and implementing lessons learned. This project leader has the following quote found

on the Internet clipped to her workstation, and reads it first thing every day. "You must communicate with your team on a personal level. You've got to be proud of your people when they win and share their disappointment when they fail. If you abandon them during hard times, they will lose faith in you; and when you have lost the faith and trust of your people, you are worthless as a leader. Leaders know their people, their families, their goals, their abilities and their dreams. You have a personal interest in their success, so you must get to know them personally."

It could be argued that the topics described above also are a significant part of building team motivation, and in a sense, that argument is true. Certainly, empowerment is a strong team motivator. This author is addressing building team motivation separately because incentives, flexible work hours, and empowerment can be also accomplished individually to address other project management challenges. Empowerment does not automatically motivate a team, it simply facilitates team motivation.

The Deming philosophy fostered by Peter Scholtes in Reference 21 was one of the basic building blocks used for building team motivation in this project. These principles of quality leadership include:

- Customer focus
- Obsession with quality
- Recognizing the structure in work
- Freedom through control
- Unity of purpose
- Looking for faults in systems
- Teamwork
- Continued education and training

Although somewhat difficult to segregate, teamwork is an important part of a quality product. This project leader believes that as the team embraces the spirit of teamwork, the members will begin to work together towards quality. Mallory states in Reference 22, "Motivation helps team members when the task at hand doesn't seem reachable or when there is conflict. It stimulates creativity and energy in team members, and contributes to a successful outcome. Usually, the results you get from a team are directly proportionate to the motivation they receive. In short, motivation serves you and the team well." Research has proven money is not the main motivator. True motivation goes beyond money. It is nonmonetary motivators that provide the big payoff. On a regular basis, this project leader reviewed and attempted to ask the questions that Mallory lists to see if she was motivating her team:

- Do I recognize each person's valuable traits and attempt to draw out these attributes?
- Do I respect each team member?
- Do I see the team as a congruent that is capable of handling the project?
- Do I participate and guide the team without being overbearing?
- Do I let the team know what the rewards are for their work?
- Do I let the team members know when I feel good about what they do and that their good work helps me too?
- Do I know each team member, his achievements and history with the company?
- Do I feel a personal commitment to the team and to the goals to be achieved?
- Do I enjoy being a team leader?

The following is a list of some of the motivators this project manager has used successfully:

- Rewarding the entire team for a big accomplishment, such as by baking a cake, bringing in doughnuts, and springing for pizza
- Hosting a social gathering for the team, such as a team picnic or team lunch
- Giving the team members decision-making powers
- Asking for the team's feedback
- Giving verbal support
- Really listening
- Conducting team warm-ups
- Setting up presentations to "showcase" what the team has accomplished
- Researching the employee recognition award system to find appropriate awards to recognize team accomplishments
- Having semi-annual one-on-one meetings with team members to determine their goals and aspirations, and then attempting to meet their objectives

- Saying thank you and meaning it both in private and at meetings with team members and management

- Writing letters of recognition

As a project manager for the government, there is no budgetary resource to use for employee motivation. The motivators supplied throughout the project came from the project leader's paycheck. However, small investments make a big payoff. The team knows it is a personal investment to show appreciation for their effort and this is special recognition that goes a long way.

5.5.1 Team Warm-ups as Motivators

Scholtes recommends team warm-ups. The purpose of these warm-ups is to get the team to leave behind concerns and ease into the meeting, to gradually focus on its task. Warm-ups are a technique generally used early in the development effort to build team motivation and cohesion.

For the project leader's first attempt at a team warm-up, questions were chosen to encourage team members to tell about their work-related backgrounds—Where did they go to school? What was their field of expertise? How did they get into this line of work? This went over quite well and people enjoyed the information exchange. After a few minutes of light conversation, a big review meeting was conducted.

On the second warm-up, each person was asked to write down names for the team. This did not go over well, and there was not much interest. On the third warm-up, the team was asked for ways the team would like to be empowered and this did not go over well either. The team seemed hesitant in airing their ideas in front of the others. The fourth warm-up, however, went well. The team leader asked questions about what parts of the project did not make sense and asked members to give one or two comments on areas they felt were at risk. Although this topic was not as light as the others, it was well received, participation was excellent, and ideas and suggestions were productive. Additional warm-ups, overall, have been successful.

In summary, occasional warm-up sessions that are somewhat personal but related to the projects, have been the most successful. Topic selection is important. If discussion centers on areas where people are uncomfortable speaking up or if they do not perceive how the warm-up will be useful to the project, the response is less favorable. Overall, the benefit of team warm-ups was most effective early in the lifecycle of development.

5.5.2 The Team Motivating Itself

One Monday morning, this project leader entered the development lab to overhear a conversation about what had been discussed at the team get-together on the previous Friday

afternoon. It appeared the team had set up its own weekly time to discuss a topic totally unrelated to work. The plan was for the presenter to rotate among the team members. At first, the project leader felt some concern about the team spending time in a non-work-related activity without approval. In a quick assessment, the project leader decided the activity was a team motivator, sanctioned it as a team-building session, and encouraged continuation of the Friday sessions. Within a couple of weeks, the project leader was actively encouraged to attend a session. The results of these meetings increased bonding of team members and in no way detracted from productivity. Performance for the rest of the Friday afternoon was much improved. Instead of winding down after lunch for the week, energy surged and work continued much longer.

The point of this example is that empowerment may take unexpected avenues as the team members grow as a team. The project leader must remain open-minded and foster the growth, rather than stifling it because it is unexpected.

5.6 PROJECT MANAGEMENT LESSONS LEARNED

It would not be practical to list all the lessons learned in a multi-year project. Those listed below are those identified as having the most potential to improve future development efforts.

- Do not promise the customer anything until it is certain the promise can be met. Software implementation does have limitations.

- Encourage team members' proactive communication that is not limited to meetings and other structured information-sharing opportunities. Developers need to listen to each other and share information when working in the lab.

- Facilitate all meetings. Come to meetings with a predefined agenda. Consistently follow effective meeting guidelines in all meetings. Document minutes and action items from all meetings.

- Ensure that every person has a backup to eliminate bottlenecks when the primary person is unavailable.

- Document and review lessons learned at the end of each major task.

- Realize that you can never get it perfect; don't be discouraged, just learn from the lessons learned, and come closer to perfection the next time.

SECTION 6

SUMMARY

What makes a successful project? No one characteristic or person dictates success, but rather a combination of many characteristics and people. The critical success factors that this project leader considers most important are:

- Structured and documented iterative lifecycle development methodology
- Implementation of software engineering repeatable processes
- Highly-motivated, technically-experienced dedicated team
- Continuous and substantial customer involvement
- Self-directed, empowered management style
- Experienced and committed project leader

Only in an ideal development environment could every principle for efficient and quality development and effective management be implemented. In today's complex development environment, even the most experienced and knowledgeable project leader will not be able to create the optimal development environment and produce error-free software. There are too many variables and unknowns in the real world. The project manager cannot control people needing emergency leave, a sudden contract change, or work stoppage because of snow. However, armed with knowledge and experience, the project manager can reduce risk and minimize problems. An empowered team, with gifted technical developers, contributes to the development of quality software.

In general, after a project ends, the team goes on to other development efforts. The customer is the ultimate judge of the quality, functionality and success of the software in supporting the business requirements. Success is generally not immediate. Customers need time to learn how the software works to use it most effectively. Also, once customers are familiar with software capabilities, they generally think of ways the software could be improved. Complete customer satisfaction is highly improbable. Because of the urgency to get this project's software deployed, the customer agreed to certain temporary workarounds. As

expected, as these workarounds are eliminated, the customer becomes more satisfied with the software.

According to experts in the field of software development, by all accounts, the project described in this document should have failed. An indication of risk is the number of new technologies incorporated in one effort. The rule of thumb is that software development efforts that tackle more than two technologies are at risk and likely to fail. This effort included the following new approaches:

- Iterative development
- Graphical User Interface
- New set of development tools
- Client-Server architecture
- Software distribution
- Rapid data transition
- Synchronization of remaining portion of the old application with the new application
- Common report interface

As documented in the previous sections of this paper, these risks and challenges were effectively overcome. Instead of a failure, the effort was a success.

In the final analysis, the team members, management, and customers determine whether the project was a development success. This project received an NSWCDL Special Merit Group Achievement Award and the AEGIS Excellence Award. The Project Leader received the N Department Leadership Award. This official recognition lends credibility to the quality of the software and the success of the project.

The team members, openly or to themselves, evaluate whether the project and the management was successful. In this project, the project manager believes the team thought the effort was a success by their requests to remain on the project even when opportunities and challenges arose. With minimal turnover, the team remained with this project leader for the next development effort. Input by team members for the awards ceremony indicated satisfaction with the project, the team, and the project leader.

This project leader deems the project successful because of the following reasons:

- There have been no Priority One changes requested, meaning the customer is able to do the required work.

- Customers have stated that the increased capabilities provided when utilizing GUI software outweigh any limitations imposed by moving from a hierarchical to a relational database.

- Metrics indicate most customer support calls are for user education or for technical support. Calls that have actually identified application defects are in the minority.

- Through an optimistic attitude, team support, and individual work efforts, the team delivered a highly complex and sophisticated system with minimal defects.

Very few large GUI applications have been successfully deployed in a client-server environment. Success in this project resulted by applying knowledge gained as a project leader in previous development efforts and the education and training received during the last four years while earning a Master of Science in Software Engineering Technical Management. It is the desire of this project leader that others may benefit from the sharing of knowledge of a successful Lifecycle Development Methodology tailored for iterative software application development.

SECTION 7

REFERENCES

1. Barker, Richard, *CASE*Method Tasks and Deliverables*, Oracle Corp., Belmont, CA 94002, Addison-Wesley Publishing Company, Reading, MA, ISBN 0-201-41697-2, 1990.

2. Barker, Richard, *CASE*Method Fast Track, A RAD Approach*, Oracle Corp., Belmont, CA 94002, Addison-Wesley Publishing Company, Reading, MA, ISBN 0-201-62432-X, 1994, pp. 2-9.

3. Dicter, Carl, "Proper Process Documents," *UNIX Review*, Vol. 12, No. 7, Jul 1994, pp. 39-43.

4. Humphrey, Watts, "Introduction to Software Process Improvement," *IEEE Technical Report*, Carnegie Mellon University, CMU/SEI-92-TR-7, Jun 1992, pp. L1-L12.

5. Paulk, Mark C., "Capability Maturity Model," *IEEE Software*, Jul 1993, pp. 24-26.

6. Paulk, Mark C., "Capability Maturity Model for Software, Version 1.1," *IEEE Software*, Carnegie Mellon University, CMU/SEI-93-TR-24, Feb 1993.

7. Olson, Timothy G., "A Software Process Framework for the SEI Capability Maturity Model: Repeatable Level," *Special Report*, Carnegie Mellon University, CMU/SEI-93-SR-07, Jun 1993, pp. SPP1-SPP27, SQA1-SQA-21.

8. Paulk, Mark C., "Key Practices of Capability Maturity Model, Version 1.1," *IEEE Software*, Carnegie Mellon University, CMU/SEI-93-TR-25, Feb 1993, pp. L24-L25.

9. Weinberg, Gerald M., *Quality Software Management*, Vol. 2, Dorset House Publishing, New York, NY, 1993, pp. 284-305.

10. Weinberg, Gerald M., *Quality Software Management*, Vol. 1, Dorset House Publishing, New York, NY, 1992, pp. 170-171.

11. *Business Week*, "Behind Oxford's Billing Nightmare," 17 Nov 1997, pp. 98-106.

12. Joiner, Brian L., *Fourth Generation Management: The New Business Consciousness*, McGraw-Hill, Inc., ISBN 0-07-0327157, NY, 1994, pp. 8-15.

REFERENCES (Continued)

13. Meredith, Jack R. and Mantel, Samuel J., *Project Management; A Managerial Approach*, John Wiley & Sons, Inc., New York, NY, ISBN 0-471-016268, 1995, pp. 121-122.

14. Wesner, John W., Hiatt, Jeffrey M.; and Trimble, David C., *Winning with Quality; Applying Principles in Product Development*, Corporate and Professional Publishing Group, Addison-Wesley Publishing Company, Reading, MA 01867, ISBN 0-201-63347-7, 1995, pp. 156-163.

15. Deming, W. Edwards, *The New Economics*, Second Edition, Massachusetts Institute of Technology Center for Advanced Engineering Study, Cambridge, MA, 1995, pp. 125-128.

16. MIL-STD-498 *Application and Reference Guidebook*, Joint Logistics Commanders, Joint Policy Coordinating Group on Computer Resources Management (located on the Internet—http://diamond.spawar.navy.mil/specs/mil-std/MIL_STD_498.html), pp. 10-11.

17. Glinow, Mary Ann Von, "Incentives for Controlling the Performance of Highly Technology and Professional Employees," University of Southern California, Graduate School of Business Administration, Los Angeles, CA 90089, 1983, pp. 5-16.

18. Wilson, Patricia, *Empowering the Self-Directed Team*, National Press Publications, Shawnee Mission, KA, 1993.

19. Juran, J. M., *A History of Managing for Quality; The Evolution, Trends and Future Directions of Managing for Quality*, ASQC Quality Press, Milwaukee, WI, 1995, pp. 643-646.

20. Vecchio, Robert P., *Organizational Behavior*, Second Edition, The Dryden Press, Orlando, FL 32887, 1991, pp. 225-227.

21. Scholtes, Peter R., *The Team Handbook: How to Use Teams to Improve Quality*, Twenty-fourth Printing, Joiner Associates Inc., Madison, WI, 1995, pp. 1-13.

22. Mallory, Charles, *Team-Building*, National Press Publications, Shawnee Mission, KA, 1991, pp. 32-38.

REFERENCES (Continued)

OTHER SUGGESTED REFERENCES

Covey, Stephen R., *The 7 Habits of Highly Effective People*, Simon & Schuster, Fireside, Rockefeller Center, New York, NY, 1989.

Humphrey, Watts S., *A Discipline for Software Engineering*, Addison-Wesley Publishing Company, 1995.

Mayhew, Deborah J., *Principles and Guidelines in Software User Interface Design*, PTR Prentice Hall, Englewood Cliffs, NJ, 1992.

Pasmore, William A., *Designing Effective Organizations, The Sociotechnical Systems Perspective*, John Wiley & Sons, 1988.

Peters, Thomas J., *In Search of Excellence*, Harper and Row, Publishers, NY, 1982.

Senge, Peter M., *The Fifth Discipline*, Double Day Currency, NY, 1990.

APPENDIX A

PROJECT PLAN EXAMPLE

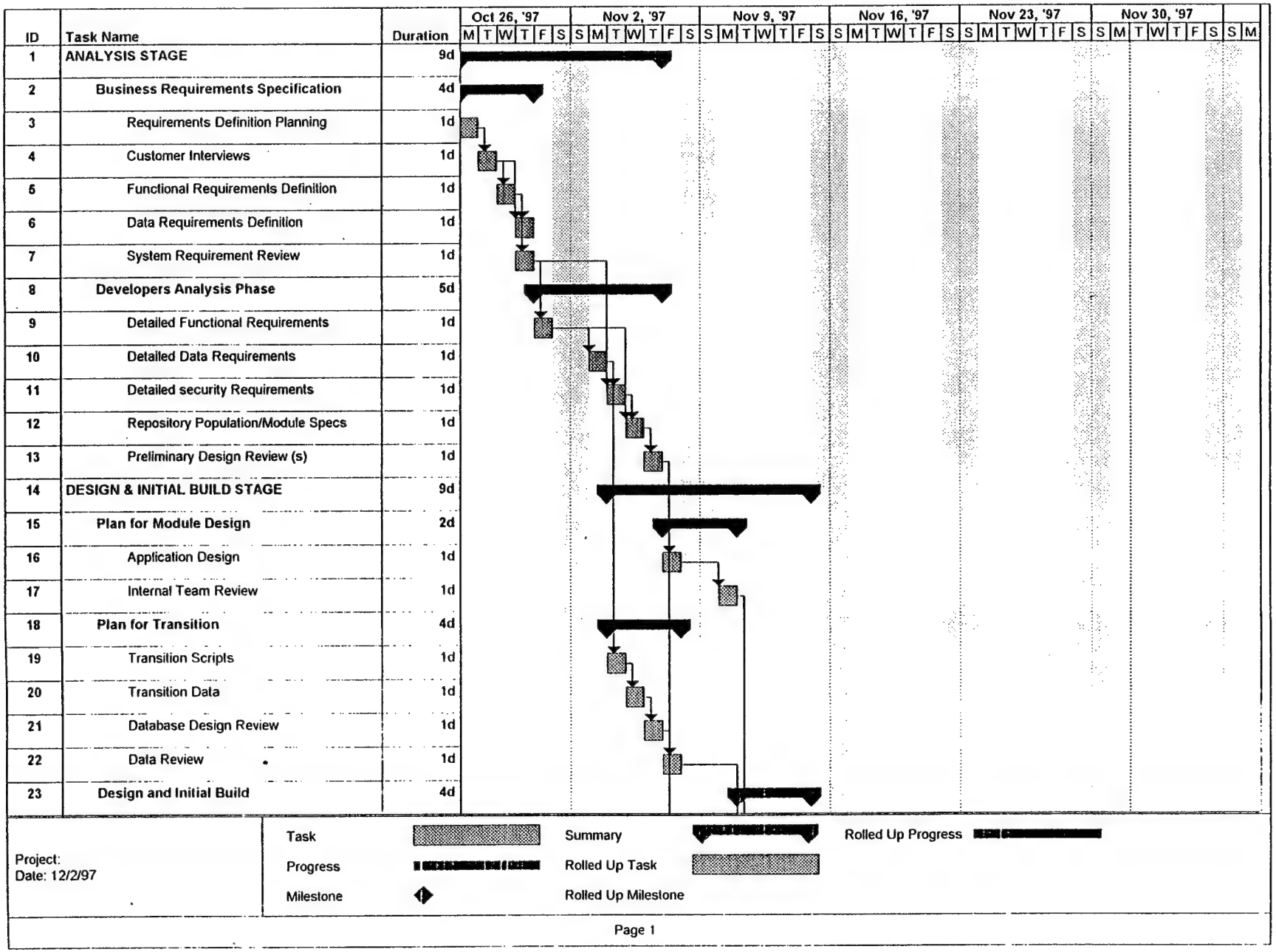

FIGURE A-1. SAMPLE PLAN OF ACTION USING LIFECYCLE DEVELOPMENT METHODOLOGY

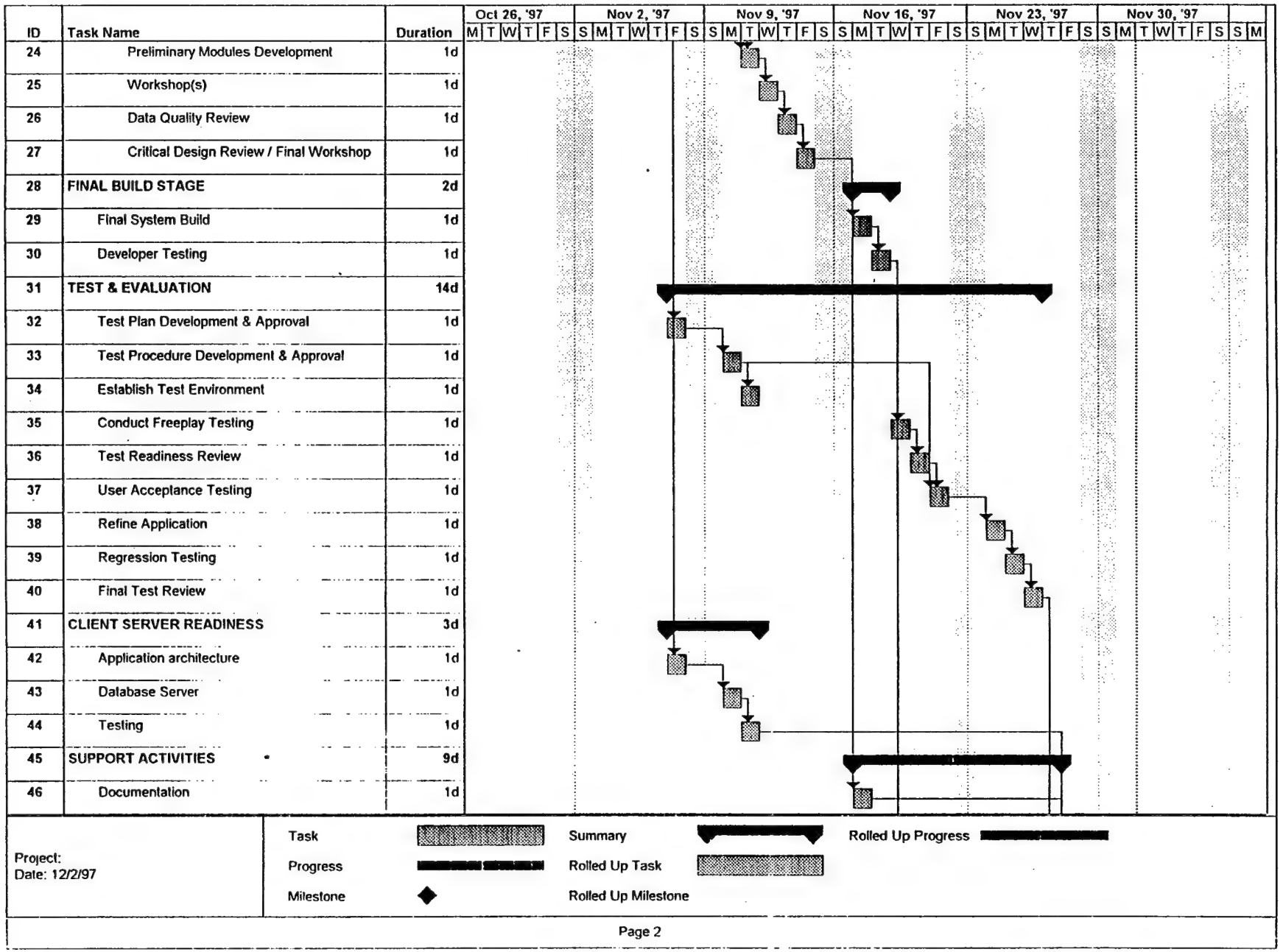

FIGURE A-1. SAMPLE PLAN OF ACTION USING LIFECYCLE DEVELOPMENT METHODOLOGY (Continued)

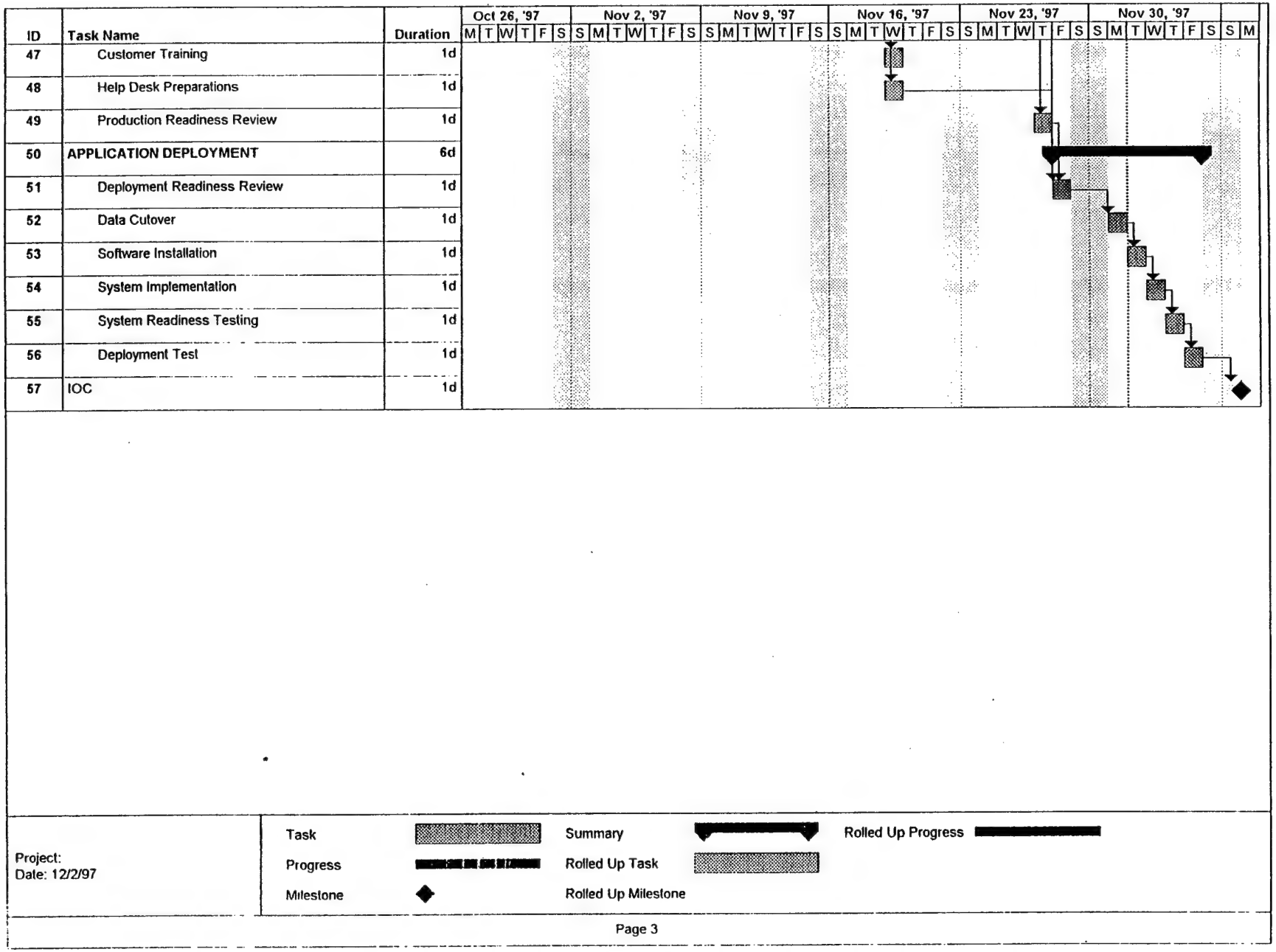

FIGURE A-1. SAMPLE PLAN OF ACTION USING LIFECYCLE DEVELOPMENT METHODOLOGY (Continued)

APPENDIX B

STATEMENT OF WORK

Statement of Work

Computer Programs Configuration Management
ACCESS Baseline 2 Development Project
October 24, 1995

This statement of work describes in a clear, concise, and non-technical language, the purpose, scope and objectives of this project. The intended audience of this document is both, the management and customer communities.

Purpose

1. The purpose of this project is to reengineer the current Computer Programs (CP) Baseline (BL) 0 application which supports the AEGIS program in the life cycle support of configuration management of computer programs. The re-engineered application will interface with AEGIS Configuration Control and Engineering Status System (ACCESS) Equipment and ACCESS Administration. This will be accomplished by developing a Graphical User Interface (GUI) client/server application which will access, transparent to the user, a distributed database architecture.

Scope of Work

1. The development team will first conduct a requirements analysis to determine the specific need as understood by the customer (N20C). High-level technical goals and objectives are identified below. From these goals and objectives, specific technical and non-technical requirements will be identified, negotiated and agreed upon as the initial set of software requirements that will be a basis for planning ACCESS BL 2. The initial set of requirements is subject to enhancement, clarification and renegotiating as the project proceeds.

2. The project manager will produce an initial project plan and schedule to perform the necessary activities to satisfy the customer's requirements. The plan and schedule is subject to refinement, adjustment and renegotiating as the project proceeds and as more accurate and more detailed information is obtained regarding the software implementation that best satisfies the customers' requirements.

3. The contractor will complete detailed analysis, transition, design, build, and test activities following a phased release approach before Initial Operational Capability (IOC) of each release of BL 2. Each phase of development will require a formal review before proceeding with the next phase. (This SOW was prepared after project initiation and as such no formal analysis data or database design reviews were conducted or is considered feasible at the time this document was prepared).

4. The effort will support data transition prior to application development with Adhoc capabilities being provided prior to the release of ACCESS BL 2. The Adhoc reporting tool used to support BL 2 deployment has been identified as Business Objects. The strategy for this development effort allows the customer to have access to Adhoc querying and reporting and provides some of the current BL 0 functionality early in the life cycle of development, rather than waiting for BL 2 IOC.

5. The scope of CP Management includes: (1) Identification of Computer Programs and Versions, (2) Tracking Documents, and (3) Managing changes, CP Baselines, and Media.

6. The following functional areas will be included in the BL 2:

- Change Control - Supports the process of creating, reviewing, maintaining, and tracking the status and implementation of Computer Program related change requests, including Computer Program Change Request (CPCR), Interface Change Request (ICR), Specification Change Request (SC), and Film Label Change Request (FLCR). In addition, it supports the various Element Change Control Boards and the AEGIS Weapons System (AWS) Change Review Board (CRB) in development of board agendas, minutes, and status reports.

- Baseline Identification - Supports the CP Baseline definition process and the configuration identification of the Computer Programs

- Training Systems - Tracking training exercise development and feedback

- Documentation Management - Supports the identification of a computer program related document and its baseline, element, and location applicability. In addition, supports the tracking of the status of the copy of a document delivered to AEGIS ship or site

Media Management - Receiving, identifying, tracking, and delivery of Media and associated documents.

7. BL 2.0 (first release of Change Control (CC)) will support the current functionality of BL 0 for the CC functional area, applicable Software Configuration Management (SWCM) related capabilities and CP administration. BL 2 will also have added functionality not available in BL 0. A detailed listing of these enhancements are enclosed as Appendix A [Appendix C in this document, MP-98/118]. Further analysis will determine if portions or all of Baseline Identification will be included in BL 2.0 or in a follow-on release.

8. Requirements to be included in future releases of BL 2 have not been adequately analyzed to include into a specific release at this time. It is assumed the second release of CC will include any remaining requirements identified for the previously defined functional areas, such as Baseline Identification. Other probable candidates that will be included in future releases include:

- Interface to STARSYS

- Seamless Electronic Bar-code Interface

- Expanded functionality in support of Life Systems Engineering Agent (LSEA)

- Expanded functionality in support of CPCR Abstract Analysis Reporting

- Other future requirements under consideration (i.e., Naval Surface Warfare Center, Dahlgren Division (NSWCDD) Test Observation Report (TOR) tracking and development and testing requirements)

Additional analysis and coordination with ACCESS Equipment and CP development will be required to determine the scope of each of these functionalities to determine to which BL 2 release they will be assigned.

9. Baseline 2.? (first release for SWCM) will support the Documentation and Media Management functions. Additional releases of CP SWCM have not been identified.

Technical Goals and Objectives

1. Exceed the functionality of the current system
2. Provide a method for the customer to respond quickly to new representations of data
3. Establish an interface with ACCESS Equipment
4. Improve capability of establishing interfaces with other systems

6. Provide an easily maintainable and well documented system
7. Provide a stable product for the CP community
8. Develop an intuitive and self documenting application
9. Reduce data redundancy
10. Improve integrity of business data via automated validation checks
11. Simplify retrieval of business information by providing more flexible querying capabilities
12. Migrate the legacy system to a GUI/Oracle environment

Other Goals and Objectives

1. Focus on customer business priorities
2. Remove barriers to information
3. Establish customer ownership
4. Deliver short term results
5. Whenever possible incorporate level two practices as described in the Software Engineering Institute's (SEI) Capability Maturity Model (CMM)

Customers

1. The senior customer representative is identified as Cathy Wood (N20), who has the authority to approve functional requirements and implementation. Two deputies will assist in this capacity (1) Jeanne Little (N21) and (2) Pam Shapiro (Synetics). Other individuals may support the decision making process; however, ultimate responsibility is directed to the above named persons. Elaine Powers will provide support for the CRB.

2. BL 2 will support the current CP customer community. A detailed list of the customers are can be obtained from the Project Manager. The locations and approximate number of these customers is identified as follows:

Location	Number of Customers
Bath	6
Port Hueneme/Corona	6
Dahlgren	143
King George	37
Pascagoula	3
Silver Springs	12

Imposed Standards

1. The development effort will follow a tailored implementation of CASE*Method Fast-Track.

2. The development effort will conform with ACCESS standards for naming conventions and dictionary usage.

3. Whenever possible, the application will emulate Microsoft Windows standards.

4. Whenever possible, the development process will strive for SEI level 2 certification.

Assigned Responsibilities

1. The Project Manager is responsible for negotiating commitments, developing the software project plan, managing the resources and activities for software development, tracking progress according to the project plan, revising the project plan as necessary, providing effective interface with sponsors, customers, and support contractors, and facilitating communication within the development team.

2. The Design Analyst is responsible for analyzing the customers' requirements and designing a technical solution that supports those needs, including transition requirements.

3. The Developer is responsible for building and unit testing the application based on the technical system design specifications.

4. The Test team members are responsible for developing the test plans and procedures, exercising the test plans and procedures, and documenting the results of the testing.

5. The Documentation team members are responsible for developing the documentation to support ACCESS BL 2. This documentation includes online and hard copy documentation.

6. The Training team members are responsible for developing the training plan, developing the training materials and conducting the training.

7. The Transition Specialist is responsible for implementing transition requirements.

8. The System Architect is responsible for developing and testing the systems architecture required to support the development and production environments.)

9. The Database Administrator is responsible for maintenance for the dictionary and models and maintaining and tuning the database)

Cost and Schedule Constraints

1. The system must be developed within the established budget as follows:

Government support	2.0 man-years (thru Jul 1996)
Government support	3.0 man-years (thru Sep 1997)
Contractor support	6.0 man-years (thru Dec 1996)
Contractor support	5.0 man-years (thru Sep 1997)

2. The system development effort will be supported by adequate support for documentation, testing and training.

3. Integration with the existing ACCESS application supporting equipment and administration.

Resource Constraints

1. The successful completion of this development effort requires adequate expertise for GUI using Oracle forms 4.5, Client-Server, Distributed Database, Business Objects, etc. The team composition must include this expertise. The government is unable to provide the expertise and as such expects to be provided this expertise from designated contractors.

2. The team is limited to the following composition:

Sharon Colston: Project Leader

Kathy Carpenter: Project Leader Backup
BL 0 Support, Analysis, Adhoc Query Tool Development

Bob Simmons: Analyst, Transition

Dawn Lundin: Analyst, Transition, CC GUI Development

Brad Skiles: Analyst, Transition, SWCM GUI Development

Jessica Louie: Analyst, Transition, DBA, GUI development

Jessica Louie: Analyst, Transition, Database Administer (DBA), GUI development

Paul Sebenik: Transition engine development, Technical deployment support

Chris Adamson: Adhoc Query Tool development, GUI development

Catherine Doughtery: GUI expertise and development for CC

Tara Carlson: Analyst, Transition, SWCM modeling, SWCM GUI development

Other Constraints

1. There is a limited number of development workstations that are available.

2. Changes to the scope must be limited, controlled and managed.

3. Integration of requested changes to the ACCESS Equipment and Administration application will support the CP BL 2 development plan.

4. Government customers are required to furnish their own hardware for utilizing CP BL 2 and the adhoc query tool. During FY96, N87 will procure and pay for the connection software and the adhoc reporting tool, Business Objects, required for all government customers, regardless of department. Government customers are required to provide their own NSWCNET connection, Ethernet Card, IP address and TCP/IP software. Plans are still being worked to address out-year software procurement support. Contractors will be required to purchase their own hardware. Contractors who wish to use BL 2 will be required to purchase their own TCP/IP and connection software. Contractors who plan to use the adhoc reporting tool will be required to purchase the adhoc reporting tool.

Critical Success Factors

1. Support and timely involvement from the CP business and customer community.

2. Identification of all key issues, assumptions and requirements.

3. Successful implementation of the minimum hardware/software requirements on client's PC.

4. Development and implementation of adequate support services which include:

   Testing
   Training
   Documentation
   Online Help
   Customer support
   Process for procurement of software
   Process for installation of software

5. Identification and prompt resolution of identified risk and potential problems.

6. Capitalizing on process improvement opportunities.

7. Experienced and well-trained GUI software developers.

APPENDIX C

DETAILED REQUIREMENTS HIERARCHY

SOFTWARE CONFIGURATION MANAGMENT (SWCM) V0 REQUIREMENTS HIERARCHY

1. The desired system provides the customers with the capability to process and track the configuration of a Computer Program as follows:

    1.1 Enforce uniqueness of a Computer Program
        1.1.1 All Computer Program names are established per the guidelines written in the CMI
        1.1.2 Validation of Computer Program to ensure uniqueness
            a. Validation occurs on the Short Program Name
            b. Given the valid Short Program Name, the system populates the long program name and AEGIS Combat System element associated with the Short Program Name from a DRS translate table which is maintained by the users
            c. Updates to a Short Program's long name and AEGIS Combat System element is made via a translate table
    1.2 Stores information that describes the configuration of the Computer Program
        1.2.1 Identifies the Computer Program
            1.2.1.1 Ensures that a valid Short Program Name is entered for each created Computer Program record
            1.2.1.2 Date of creation/update is system generated
            1.2.1.3 Deletion of computer program records requires deletion of all "children" records referencing the computer program
            1.2.1.4 Creation/Updates/Deletions are restricted to users with Status Accounting privileges
            1.2.1.5 Capability to chain to the NSWCDD ID application
        1.2.2 Defines the version control of the Computer Program
            1.2.2.1 Files
                1.2.2.1.1 Upon creation of file records, system will add patch records for all patch file types except those starting with the word "patch" and do not contain the word "map". The following fields are populated with the data from the Files record: patch name, patch directory, patch type, and creation date
                1.2.2.1.2 Ensures the following information is entered for every File record: Short Program Name, version, NSWCDD ID number, Patch/file name, Directory, and File type
                1.2.2.1.3 When copying a Files record to another short program name, version, and NSWCDD ID, ensures that the short program and version exists for that NSWCDD ID number
                1.2.2.1.4 Validate the following fields for all Files records: Short program name, File type, and sub short program name
                1.2.2.1.5 Creation/updates/deletion are restricted to those with Status Accounting privileges
                1.2.2.1.6 Capability to chain to Patch Detail and NSWCDD ID applications
            1.2.2.2 Patch Detail
                1.2.2.2.1 Ensures the following information is entered for every Patch Detail record: Patch name, patch directory name, and patch type

1.2.2.2.2 Ensures that the patch type entered for the Patch Detail record is the same as the file type for the matching File record. If not, the system will update the file type with the value entered in the patch type

1.2.2.2.3 Creation/updates/deletion are restricted to those with Status Accounting privileges

1.2.2.2.4 Date of creation/update is system generated

1.2.2.2.5 Deletion of Patch detail records requires deletion of all Patched-in changes and files records referencing the patch detail record

1.2.2.2.6 Capability to chain to Files and Patched-in Changes applications

1.2.2.3 Patched CPCRs

1.2.2.3.1 Ensures that the following information is entered for each Patched CPCR record: patch name, patch directory, CPCR number, and baseline

1.2.2.3.2 Ensures that only patched CPCR records are created for a valid baseline

1.2.2.3.3 Ensures that only patched CPCR records are created for those that have a Patch Detail record. If there is no patch detail record, system prompts you to enter a valid patch type (restricted list of values) and then creates the patch detail record

1.2.2.3.4 Date of creation/update is system generated

1.2.2.3.5 Creation/update/deletion is restricted to those with Status Accounting privileges

1.2.2.3.6 Copy functionality allows Patched CPCR records to be copied from one baseline to another baseline. Ensures that patched CPCR records are only copied to a valid baseline

1.2.2.3.7 Copy functionality displays how many CPCRs were copies from the old baseline to the new baseline

1.2.2.3.8 Capability to chain to Patch Detail application

1.2.2.4 Sourced CPCRs

1.2.2.4.1 Ensures that the following information is entered for each Sourced CPCR record: short program name, version, CPCR number and baseline

1.2.2.4.2 Ensures that only Sourced CPCR records are created for valid baselines

1.2.2.4.3 Ensures that a valid ORTs type is entered for ORTs short program names

1.2.2.4.4 Validates the following fields: short program name, ORTs type

1.2.2.4.5 Copy functionality allows ORTs Sourced CPCR records to be copied from one baseline to another baseline. Ensures that sourced CPCR records are only copied to a valid baseline

1.2.2.4.6 Copy functionality displays how many CPCRs were copied from the old baseline to the new baseline

1.2.2.4.7 Date of creation/update is system generated

1.2.2.4.8 Creation/update/deletion is restricted to those with Status Accounting privileges

1.2.2.4.9 Capability to chain to NSWCDD ID application

1.2.2.5 NSWCDD ID Information

1.2.2.5.1 Ensures that the following information is entered for each NSWCDD ID record: Short program name, version, and NSWCDD ID number

1.2.2.5.2 Validate the short program name

1.2.2.5.3 System populate the Element from entry of or modified of short program name

1.2.2.5.4 Creation/update date is system generated

1.2.2.5.5 Creation/update/deletion is restricted to those with Status Accounting privileges

1.2.2.5.6 Deletion of NSWCDD ID Information requires deletion of all files, inventory, and version "children" records linked to that NSWCDD ID record

1.2.2.5.7 Capability to chain to Files, Computer Program, and Inventory applications

1.2.2.6 Inventory Information
1.2.2.6.1 Ensures that the following information is entered for each Inventory record: Short program name, version, NSWCDD ID number, and ship/site
1.2.2.6.2 Validate the short program name
1.2.2.6.3 System populates the Element from entry of or modified of short program name
1.2.2.6.4 Creation/update date is system generated
1.2.2.6.5 Creation/update/deletion is restricted to those with Status Accounting privileges
1.2.2.6.6 Copy capability allows user to copy inventory records to another ship/site and baseline
1.2.2.6.7 Copy capability has three choices: AWS only, NONMK7 only, ACTSS only
1.2.2.6.8 Ensures that only production inventory records are copied
1.2.2.6.9 If the ship/site and baseline copying to has not been created, system creates the ship inventory record
1.2.2.6.10 When copying ACTSS, ensures that the element contains "ACTS" and the long name of contains "Exercise scenarios" or "multiscenario"
1.2.2.6.11 When copying AWS, ensures that element containing "ACTS" and the long name not containing "Exercise scenarios" or "multiscenario" are included. When copying inventory records, if the program and version exists on the ship/site and baseline copying to, the system will check to see if data exists in the SHIPPED and DATE RECEIVED fields. If data exist, the system will retrieve the earliest date the program/version was shipped and copy that information to the new ship/site and baseline inventory record created. If no data exists in those two fields, nothing is copied to the new inventory records
1.2.2.6.12 Restricts the values for SHIP DISK as follows: For Baseline 4 and 5: PMA, COM, or SEN. For all other baselines: PMA, ADS, TAC
1.2.2.6.13 Capability to chain to NSWCDD ID and Ship Inventory applications

1.2.3 Defines the ship/site baseline configuration of the Computer Program
1.2.3.1 Ensures that the following information is entered for each ship/site and baseline information: ship/site and baseline
1.2.3.2 Validates the ship/site
1.2.3.3 Creation/update date is system populated
1.2.3.4 Creation/update/deletion is restricted to those with Status Accounting privileges
1.2.3.5 Deletion of Ship/site baseline information requires deletion of related inventory records
1.2.3.6 Headers of DDD, master, shipboard, and TDS tapes are restricted as follow: For Baselines 4 and 5 - Sensor and Weapons, Command and Display, PMA. For all other baselines - ADS, Tactical, PMA

1.3 Retrieve information that describes the configuration of a Computer Program
1.3.1 Produce the Master Parts List (is not automated in Baseline 0. Is produced manually by CM)
1.3.2 Generate Computer Program Receipt Logs which documents each piece media for a specific computer program is already resides on the ship/site or will be delivered to that ship/site for a specific baseline
1.3.3 Generate Disk Description Documents for each master disk pack that is being delivered to the ship/site for a particular baseline
1.3.4 Generate Load Program Documents for each computer program that is being delivered to the ship/site for a particular baseline

2. The desired system provides the customers with the capability to process and track media information received at the AEGIS Computer Center or delivered to AEGIS ship/sites as follows:

2.1 Process AWS media
2.1.1 Stores information for processing AWS media

2.1.1.1 Receipt of AWS media
2.1.1.1.1 Ensures that only when Aegis Weapon system is populated with the value "N" will AWS be allowed to use the "marked for" screen
2.1.1.1.2 Creation/update date for the receipt screens and the marked for screen are system generated
2.1.1.1.3 Creation/update/deletion is restricted to those users with AWS privileges
2.1.1.1.4 Ensures that the following fields are entered upon creation of a receipt record: receipt number, media id number, classification (except when media status is obsolete), originator, element, media type, short program name (if a version exists), short program name2 (if a version2 exists)
2.1.1.1.5 Ensures that the following fields are entered upon creation of a 'marked for' record receipt number, media id number, copy number, and marked for location
2.1.1.1.6 Ensures that a '*' is displayed by the media number on the listing screen if media status is populated with any valid value other than 'retained'
2.1.1.2 Delivery of AWS media
2.1.2 Retrieve AWS media information
2.1.2.1 Transmittal Requests
2.1.2.2 Receipt Forms
2.1.2.3 Receipt Logs
2.1.2.4 ATL Log Report
2.2 Process NON-MK7 media
2.2.1 Stores information for processing NON-MK7 media
2.2.1.1 Receipt of NON-MK7 media
2.2.1.1.1 Ensures that Aegis Weapon System is populated with the value "N" by the system
2.2.1.1.2 Creation/update date for the receipt screen and the marked for screen are system generated
2.2.1.1.3 Creation/update/deletion is restricted to those users with NONMK7 CM privileges
2.2.1.1.4 The following fields are skipped on the receipt screen: listings, AEGIS weapon generated, baseline, NSWCDD request letter, incoming media, vendor ident number, qa stamps, source baseline, patch set, pack ident number, function code
2.2.1.1.5 The following fields are skipped on the second element receipt screen: source baseline2 and patch set2
2.2.1.1.6 Ensures that the following fields are entered upon creation of a receipt record: receipt number, media id number, originator, classification (except when media status is obsolete), element, media type, short program name (if version exists), and short program name2 (if version2 exists)
2.2.1.1.7 Ensures that the following fields are entered upon creation of a "marked for" record: receipt number, media id number, copy number, and marked for location
2.2.1.1.8 Ensures that a '*' is display by the media number on the listing screen if media status is populated with any valid value other than "retained"
2.2.1.2 Delivery of NON-MK7 media
2.2.2 Retrieve NON-MK7 media
2.2.2.1 Transmittal Request
2.2.2.2 Receipts Forms
2.2.2.3 Purge Reports
2.2.2.4 ICL Receipt Log
2.2.2.5 ATL Log Report
2.3 Process User media
2.3.1 Stores information for processing User Media
2.3.1.1 Ensures the following information is entered for each user media record created: media number, classification, user name, and user group

2.3.1.2 Ensures that these fields are available only for TAPE media: incoming number, shipped from, received date, and user code.
2.3.1.3 Ensures that these fields are available only for DISK media: baseline, serial number, date last modified, and ship/site
2.3.1.4 Validates user name, media type, media classification, alpha character preceding each user media
2.3.1.5 Creation/update/deletion is restricted to those users with MAD privileges
2.3.1.6 Creation/update date is system populated
2.3.1.7 Copy capability restricted to only copying "freed" media from one media number to the next media number
2.3.1.8 When "freeing" a TAPE media, the title field is populated with the verbose 'Free' and classification is populated with the letter "U". All other fields with the exception of the media number, creation/update date, and creation/update username are null out
2.3.1.9 When "replacing" a DISK media, the title field is populated with the verbose "to be replaced". All other fields with the exception of the media number, creation/update date, creation/update username and media type are null out
2.3.1.10 When "releasing" a DISK media, the title field is populated with the verbose "not assigned", classification is populated with the letter "U", and the serial number and barcode number are retained. All other fields with the exception of the media number, media type, creation/update date, and creation/update username are null out
2.3.1.11 Media number are packed with leading zeroes
2.3.2 Retrieves User Media
2.3.2.1 Generate TAPE media reports
2.3.2.1.1 Generate tape media "Free" report which prints those media numbers that are available for checkout
2.3.2.1.2 Generate tape media report of all media numbers assigned to users
2.3.2.1.3 Generate tape media report of all media numbers assigned to a specific user
2.3.2.1.4 Generate tape media report of all media received from outside sources
2.3.2.2 Generate Disk media reports
2.3.2.2.1 Generate disk media "Not assigned" report which prints those media numbers that are available for checkout
2.3.2.2.2 Generate disk media report for all disk numbers assigned to users
2.3.2.2.3 Generate disk media "carrying case" report
2.3.2.3 Generate other media reports
2.3.2.3.1 Generate barcode report

3. The desired system provides the customers with the capabilities to process and track documentation media delivered to AEGIS ship/site as follows:
3.1 Process the Master Documents
3.1.1 Store information that describes the master document
3.1.1.1 Ensures that the following information is entered for each master document record created: document number, document type, document title, classification, document date
3.1.1.2 Validates stage, classification, document status
3.1.1.3 Restricted list of values for AWS field
3.1.1.4 Ensures that if a stage date is entered, a stage must be entered
3.1.1.5 Ensures that if a change stage date is entered, a change stage must be entered
3.1.1.6 Ensures that if a change date is entered, a change number must be entered
3.1.1.7 Capability to add multiple master documents to the same receipt number
3.1.1.8 Creation/update date is system populated
3.1.1.9 Creation/update/deletion is restricted to those with document privileges
3.1.1.10 Capability to chain to Document Delivery
3.2 Process the Deliverable Documents
3.2.1 Store information that describes the delivery of documents

3.2.1.1 Ensures that the following fields are entered for each delivery document record created: document number, ship/site, baseline, number of copies, document form and element

3.2.1.2 Auto populates the following fields if "chain" from an populated master document screen: accession number, document number, document date, change number, change date, stage, stage date, change stage, change stage date, incorporated, part number, revision level, and volume number

3.2.1.3 Creation/update date is system generated

3.2.1.4 Creation/update/deletion is restricted to those with document privileges

3.2.1.5 Validation on ship/site, baseline, document delivery status and element

3.2.1.6 Restricted list of values for document form

3.2.1.7 Ensures that a master document record exists before creation of delivery document is allowed

3.2.1.8 Ensures that if a change date is entered, a change number must be entered

3.2.1.9 Ensures that if a stage date is entered, a stage must be entered

3.2.1.10 Ensures that if a change stage date is entered, a change stage must be entered

3.2.1.11 Ensures that if Received date has been populated, then "Y" is move to Receipt verified

3.2.2 Retrieve document information

3.2.2.1 Generate document enclosure report from the master document records

3.2.2.2 Generate document receipt report from the master document records

Generate document transmittal report from the document delivery records

APPENDIX D

RISK AND CONTINGENCY PLAN

Purpose

The execution of a complex software engineering project entails risk, and the evidence of prolific project failure in software engineering indicates risk management is not a refined discipline for most software engineering project managers. This plan defines a risk management paradigm adapted from the work of the Software Engineering Institute, identifies risk areas, and provides continuing updates on actions taken to address risk.

Defining Risk

Frequently project managers and teams track issues, tasks, and risks in an uncoordinated manner. The process of defining risk in this plan develops risk areas by creating a comprehensive list of project issues and converts those issues into project-level or task-level risks. These risks are then communicated to appropriate individuals and addressed.

A risk is defined as an event that may occur in the course of a project that has an adverse impact on the cost, schedule, or quality constraints defined for that project. At the task level, a risk is an event that may occur in the course of completing the task that has an adverse impact on the cost, schedule, or quality of the task's products.

Therefore, the definition of the project and task constraints are critical in defining and prioritizing risk. It is also important to understand the probability component of risk; a specific risk may not occur in the course of the project. Project team turnover may be identified as a project risk with substantial impact on cost, schedule, and quality, but the turnover itself may not occur.

The project manager's job is to identify, communicate, analyze, and address risks at the project level, and oversee task level risk management. The project team identifies, communicates, analyzes, and addresses risks at the task level.

Project-Level Risk

Project-level risks represent substantial areas of concern that cross task boundaries. These risks often require involvement of the customers to resolve. Customer involvement at the project level is key because the project constraints in jeopardy (cost, schedule, and quality) are all determined at the discretion of the customer.

These constraints represent the basic investment of resources (cost, schedule) the customer has committed for a specific return (quality). Risks at the project level represent the items that will most likely impact the relationship between investment and return, and should be closely scrutinized.

Task-Level Risk

Task-level risks represent areas of concern to the project team in executing specific tasks that contribute to the overall project's outcome. While not as visible to the customer community, addressing these risks is imperative to meeting overall constraints.

By defining risks at the task level, team members are able to clearly understand the complexities of their tasks and prioritize their actions to ensure successful task completion.

Risk Management Paradigm

The paradigm used in managing risk at both the project and task levels is consistent. Fundamental to the paradigm is the recognition that elimination of all risk is unlikely, and therefore resources should be applied to higher-impact risks first. At the project level, customers must be involved in the prioritization process. Tables D-1 and D-2 presents an overview of the risk management process used in this plan.

Project-Level Risk Management

Identify—Project-level risks are identified, described, and evaluated versus cost, schedule, and quality constraints. The likelihood of a risk occurring is determined, and risks are prioritized by a joint customer/project manager review.

Analyze—Once prioritized, risks are analyzed to identify likely actions to reduce or eliminate the risk. The cost of elimination or mitigation are weighed against the likelihood of the risk occurring and the cost to the program if the risk is accepted. If accepted, the risk is continually tracked to ensure the assumptions about impact to constraints does not change as more information is gathered.

Plan—If the choice of the project manager is to eliminate or mitigate the risk through actions, these actions must be converted to tasks and included in the project plan. These tasks are then tracked as executed to ensure the planned elimination or mitigation occurs.

Track—Accepted risks, as well as those targeted for mitigation or elimination, are continually tracked as the project progresses.

Task-Level Risk Management

Identify—Task-level risks are identified, described, and evaluated versus cost, schedule, and quality constraints in the context of the task itself. The likelihood of a risk occurring is determined, and risks are prioritized by the team members responsible for the task.

Analyze—Once prioritized, risks are analyzed to identify likely actions to reduce or eliminate the risk. The cost of elimination or mitigation are weighed against the likelihood of the risk occurring and the cost to the program if the risk is accepted. If accepted, the risk is continually tracked to ensure the assumptions about impact to constraints does not change as more information is gathered.

Plan—If the choice of the project manager is to eliminate or mitigate the risk through actions, these actions must be converted to tasks and included in the specific task plan. These tasks are then tracked as executed to ensure the planned elimination or mitigation occurs.

Track—Accepted risks, as well as those targeted for mitigation or elimination, are continually tracked as the project progresses. Team members should provide periodic reporting of risk management activities.

TABLE D-1. ACCESS COMPUTER PROGRAM MANAGEMENT PROJECT-LEVEL RISKS

The following risks have been identified at the project level as of May 1, 1996.

Risk	Responsibility	Likelihood	Schedule Impact	Cost Impact	Quality Impact	Status	Contingency Option
Software CM procedures not defined for N87 deployment and maintenance of Client products	Sharon Colston	25%	**High**	Low	**High**	Working	Delay IOC
*Customer Support tasks not completed by IOC for BL 2	Sue Straughan	50%	**High**	Low	**High**	Working	1) Delay IOC; 2) Reduce desired support
*Integration and Site Testing Resources Not Available from N87	Sue Straughan	25%	**High**	**High**	**High**	Working	1) Delay IOC; 2) Address as performance issues arise; 3) Live with undesired results
*Delivery date expectations of customers not reflective of reality	Sharon Colston	25%	**High**	Low	**High**	Working	Improve communications with customers
*Impact of new environment on system functions such as E-mail not resolved prior to IOC	Dennis Smith	25%	Moderate	Moderate	**High**	Working	Provide reduced or temporary solution
User Acceptance testing approach not developed and supported	Sharon Colston	10%	**High**	Moderate	**High**	Working	1) Delay IOC; 2) Assume product will have defects
Continued technical difficulties impact schedule adversely	Sharon Colston	50%	**High**	Moderate	**High**	Working	IOC could be impacted

* Needs to be resolved by N87 management

TABLE D-1. ACCESS COMPUTER PROGRAM MANAGEMENT PROJECT-LEVEL RISKS (Continued)

The following risks have been identified at the project level as of May 1, 1996.

Risk	Responsi-bility	Likeli-hood	Schedule Impact	Cost Impact	Quality Impact	Status	Contingency Option
*Customer Training tasks consume substantial project resources	Sharon Colston	25%	Moderate	Low	Moderate	Analyze	1) Provide needed training at schedule impact; 2) Delay IOC for training; 3) Expect customers to use without training
SWCM development cannot proceed because of inadequate resources	Sharon Colston	50%	**High**	Moderate	Moderate	SWCM remains behind plans. May be able to catch up at later time	
Approach for CSR function not defined prior to Detailed Design tasks	Sharon Colston	10%	Moderate	Moderate	Low	Requirements have been submitted to N87 ACCESS Administration	Proceed and resolve when possible; understand there may be design and schedule impact
*Customer Documentation tasks consume substantial project resources	Sharon Colston	25%	Moderate	Low	Moderate	Analyze	1) Provide reduced documentation; 2) Acquire additional resources

* Needs to be resolved by N87 management

TABLE D-2. ACCESS COMPUTER PROGRAM MANAGEMENT TASK-LEVEL RISKS

The following risks have been identified at the project level as of May 1, 1996.

Risk	Task	Responsi-bility	Likeli-hood	Schedule Impact	Cost Impact	Quality Impact	Status	Contingency Option
*Requirements documentation is not properly managed prior to IOC	Design/ Build	Sharon Colston	25%	Low	Low	Moderate	Working	1) Develop web/home page to assist in management; 2) Develop manual process to support
Detailed process definition for detailed design and build task not completed prior to beginning of detailed design task.	Module Redesign Planning	Sharon Colston	10%	**High**	Moderate	Moderate	Process under development	Proceed before process is in place; assume some redesign or schedule impact may result
User understanding and acceptance of relational technology not provided	Test	Sharon Colston	75%	Moderate	Moderate	**High**	Analyze	
*Production/architecture environment not in place prior to build	Build Testing	Dennis Smith	50%	**High**	Low	Moderate	Plan	1) Delay IOC; 2)Use development platform for production and return to HP for development; will have schedule and quality impact
*Performance requirements support not available or acceptable	Design	Dennis Smith	75%	**High**	**High**	**High**	Analyze	Live with the outcome of not assessing
Synchronization between CP and SWCM is not possible, or requires greater effort than planned.	SWCM Transition	Paul Sebenick	25%	**High**	High	**High**	Analyze	1) Delay IOC; 2) Stop other tasks to complete and adjust schedules accordingly
Operational production strategy is more complex than anticipated	DBA Support Transition	Paul Sebenick	25%	Moderate	Moderate	Moderate	Analyze	1) Assign additional N87resources to support; 2) Delay IOC

* Needs to be resolved by N87 management

TABLE D-2. ACCESS COMPUTER PROGRAM MANAGEMENT TASK-LEVEL RISKS (Continued)

The following risks have been identified at the project level as of May 1, 1996.

Risk	Task	Responsi-bility	Likeli-hood	Schedule Impact	Cost Impact	Quality Impact	Status	Contingency Option
Implementation of role-based security is more time consuming than planned (DBA level)	Unknown	Paul Sebenick	25%	Moderate	Low	Low	Analyze	1) Adjust schedule; 2) Reduce requirements
BO Batch capabilities	Design	Paul Sebenick	25%	**High**	Low	**High**	Analyze	Eliminate requirement
User enrollment process not defined	Design	Paul Sebenick	25%	**High**	Low	**High**	Analyze	1) Delay IOC
BO V4 upgrade not planned	Design	Paul Sebenick	50%	Low	Low	Low	Analyze	
Reports lag integration	Build	Jessica Louie	50%	**High**	Low	Moderate	Working	
Failure to complete post workshop activities	Design/ Build	Team	50%	Moderate	Low	Moderate	Working	
Failure to track and complete outstanding changes and APERS	Design/ Build	Team	75%	**High**	Low	**High**	Working	
DSIR requirements not signed off	Unknown	Bob Simmons	10%	Low	Low	Moderate	Analyze	Leave open with no customer closure
Report interface for CP Management is not included in cost, schedule estimates and cannot be absorbed	Conduct Redesign	Delmar Lester	25%	Moderate	Moderate	Low	Analyze and Plan	1) Eliminate interface, expect customers to have difficulty using; 2) Extend schedule if necessary
BL0 data archiving not complete prior to IOC	Unknown	Kathy Carpenter	25%	Low	Low	Low	Analyze	1) Adjust schedule; 2) Reduce requirements; 3) Delay IOC
BO CM procedures not in place by IOC	Build	Kathy Carpenter	25%	Low	Low	Moderate	Analyze	1) Delay IOC; 2) Implement without formal control and expect cleanup issues and schedule impact

APPENDIX E

TESTING AGREEMENT MEMORANDUM

02 April 1997

From: Cathy Wood, N20C
To: Element CP TA's
Subj: Upcoming testing and release of ACCESS BL 2

As we near Initial Operating Capability for ACCESS BL 2, Computer Program Change Control, we enter possibly the most critical phase, User Acceptance Testing (UAT). In order to deliver a system that meets the functional business needs of the customer, intensive testing must be conducted. The purpose of this memo is to notify all TA's of the upcoming ACCESS BL 2 UAT, to present the testing strategy, and to obtain your support and commitment to the testing strategy.

This testing, using a set of user approved test procedures, will verify data security, role security and acceptable functionality of BL 2 modules in support of the CP element business process.

In order to ensure understanding and commitment for this critical activity, an agreement between the Aegis CP Element customer community and the N87 development team has been drafted. A copy of this agreement is attached. (Attachment 1)

The N87 development team has worked closely with the customer to minimize the impact of testing on the community. The testing schedule has been designed to allow the customer some flexibility in when testing can be performed during the month of May. Details are provided in the agreement, however the following summary is presented for convenience. A select group of customers have been designated as testers. These people were selected based on their knowledge of the business and the criticality of their input to the ongoing effort. Most of them have been involved in previous analysis and design reviews and this is their opportunity to ensure the requirements have been implemented correctly.

The core team will consist of twelve testers and six alternates. The alternates are to be used only when the core tester is absent from work or unavoidable circumstances. It is imperative the commitment and continuity in UAT be supported. In order to support each of you during UAT, I promise to personally limit additional change control tasking requests during May to allow the testers time to commit to the testing schedule.

A list of requested testers is as follows:

Wendi Dennison	DO 28	C. Bucholdt
Debbie Shelkey-Green	DO 27	C. Caldwell
Theba Allen	DO 23	V. Basuino
Sherri Barker	DO 25	L. George
	DO 29	E. Clarkston
	DO 30	G. Heberlein
Kim Smith	DO 30	G. Herberlein
Pam Cole	DO 29	E. Clarkston
Barbara Newton	DO 26	E. Amburgey

As mentioned previously, the schedule is designed to support the customer. Customers will be expected to train on Friday, 28 April and test every Friday beginning 2 May through 30 May. In addition to the mandatory testing on Fridays that will be held at Synetics, testers will be expected to complete the unfinished tests in the N87 Lab at their convenience on other days of the week for a few additional hours. A copy of the testing schedule is also attached (Attachment 2).

In order to prepare for testing, 28 April has been designated as a mandatory day for training. During the morning, Windows '95 training will be conducted. In the afternoon, a testing overview and introduction to the test procedures is planned. Following this approach, testers will be prepared to begin testing on 2 May.

It is requested that you respond by 11 April to indicate your agreement with the testers that have been identified and your support for the UAT commitment. In addition, should you desire to have any additional persons be part of the core test team, there are six additional vacancies. This commitment should not be taken lightly as these additional people are expected to have the same level of commitment and indeed, more so, as they will not be part of the mandatory Friday testing and will test only by scheduling time in the N87 Lab.

Thank you for your support. Please contact Sharon Colston (scolsto@nswc.navy.mil, 653-7047) with the above requested information.

Attachment 1

Computer Programs Management Change Control Customer
and Developer Testing Agreement
26 March 1997

COMMITMENT

Customers are expected to commit to the following expectations which will be documented in an agreement between the N87 developer test team and the AEGIS CP Element customer community.

1. Testers are expected to be committed to all the test requirements, so it is strongly recommended that customers do not volunteer to be testers if not planning to commit the time and energy necessary to complete all testing requirements.
2. The sponsor, Cathy Wood, is expected to support testing as a high priority for those identified to fill that role.
3. If the customer does not adhere to the testing schedule, then the schedule may be impacted or the software may be released at risk.
4. There will be a core group of not less than 12 customers dedicated to testing which will test the following roles: CRB, Element, Reviewer, General User (originator only), General User (other), and SPOC. (ATT, System Test, and Fleet Support are considered special roles and will be handled individually with lab time dedicated to testing with these customers.) There should be alternates designated by the customer for each core tester (for a maximum of 18 testers).
5. Up to 6 additional testers may be added to the 12 required testers (for a maximum of 18 testers) at the customer's request; however, they will be expected to be committed in the

same manner as the core group of testers. It is in customer's best interest to have more than 12 testers; however the additional testers will have to be more flexible in when and where they test.

6. Alternate testers will only be used if a core tester is not available.
7. Every tester will complete his/her entire individualized test package regardless of element or role.
8. These testers or their alternates will have the authority to sign off approval for each procedure.
9. The collective approval of all test procedures and final approval from Cathy Wood will successfully conclude UAT.
10. Changes from UAT will be documented by APERs.
11. Changes required by the testing will be signed off/approved on a case by case basis. If substantial changes are required early on in testing, testing may be suspended and restarted at a later time.
12. The original testers will be required to do their own re-testing.
13. Reports will be included in testing whenever possible. Those not available will be tested in follow-on UAT sessions for customer sign off approval.

Attachment 2

SCHEDULE

1. The UAT testing will be conducted the month of May and will include module functional testing, system testing and integration testing. Optional testing will be available upon customer request after the intensive structured testing is completed.
2. The UAT testing has been scheduled for a month, but may take less time than traditional UAT since test scripts are GUI-oriented and testing may be completed by customized tester scheduling.
3. The core 12 testers will be expected to test all day Friday beginning 2 May and any additional hours at other times during the week to complete the test package.
4. The lab will be open Monday afternoon and all day Tuesday for customer sign-up for one of the six available workstations
5. The lab will be available for testing on Wednesdays and Thursdays, as well, by special appointment; this is just a planning formality imposed to allow blocks of time to be available for other requirements, such as report workshops.
6. Additional testers may sign up for time on space availability in the lab.
7. A copy of the test schedule will be developed and posted in the lab and at Synetics. Testing on Fridays at Synetics is mandatory.
8. There will be a two hour mandatory training meeting the week of 28 April for all testers (date to be provided by P. Shapiro).
9. Testers will work at their own pace; however they will be required to keep pace with the testing schedule. In this manner they can be a little more flexible with their own schedule.

10. UAT is expected to be completed by 30 May in order to attempt to make a third week of July IOC

11. Monday mornings will be dedicated to developer review of the testing. No tester involvement is required.

APPENDIX F

DATA TRANSITION REVIEW PROCEDURES AND CHECKSHEETS

Data Transition Requirements Review

Purpose

1. To **detect** and **remove**
   defects from the **data transition requirements early** and **efficiently**.

2. To develop
   a better **understanding** of the **data transition requirements** and
   defects that might be **prevented**.

3. To achieve
   more **predictable quality** in the **data transition** than can be achieved
   without a review.

4. To provide project **management**
   with **reliable information** indicating **progress** in the software development
   project by **assessing** the **data transition requirements** for **fitness for use**
   in **subsequent** data transition activities.

Activities

1. **Prepare** for Data Transition Requirements Review.

2. **Conduct** Data Transition Requirements Review.

3. **Debrief Results** from Data Transition Requirements Review.

Prepare for
Data Transition Requirements Review

Objective

To enable
   review **participants** to **conduct** a data transition requirements **review**
with maximum **effectiveness.**

Activities

1. The **producers** of the data transition requirements that will undergo review **verify** that the requirements are **ready** for review. The data transition requirements are represented by tailored reports containing excerpts from the Data Sourcing Information Repository (DSIR), PL/SQL transcripts, and tables of target data.

2. The software project manager selects **participants** for the review to provide appropriate **coverage** of the data transition requirements that will undergo review.

3. The software project manager assigns **roles** (leader, recorder, reviewers) to the participants.

4. The review leader **queries** the participants to determine their **availability** for the review.

5. The review leader reserves a **location** for the review and reserves all necessary **resources** for the review.

6. The review leader **gathers** all necessary review **materials**.

7. The review leader posts a **notice** to the participants that specifies the **time** and **place** for the review and identifies the data transition requirements that will undergo review.

8. The review leader **distributes** an agenda and copies of pertinent review **materials** to the participants at least 3 to 5 days before the review.

9. The review leader **queries** all participants at least one day before the review to ensure that

   each participant remains **available** for the review and
   each participant is adequately **prepared** for the review.

Conduct
Data Transition Requirements Review

Objectives

1. To **ensure**
   all **source data** are mapped to data targets or accounted for via some resolution.

2. To **ensure**
   all **target data** are mapped to data sources or accounted for via some resolution.

3. To **ensure**
   the **contents** of the **Data Sourcing Information Repository** (DSIR) have been accurately **translated** into the **rules database** which contains input for the transition engine.

Activities

1. The **review leader** initiates the review by identifying each participant by **role** and ensures that all planned participants are present. All **participants** affirm their readiness to proceed.

2. The **review leader** briefs the group on the **review objectives sequence** of items scheduled start and completion times **logistics** information.

3. The **review leader** and a **reviewer** initiate the review by walking through all steps for a **target table** in front of the group. The review leader ensures that all reviewers understand

the **conceptual basis** for each **check point**
the **context** and **criteria** for passing or failing the check point
how to record **issues** and **related issues**

Ad-hoc training is conducted as required to ensure that all reviewers are ready to proceed.

4. The **reviewers**, working as a team, review all data transition requirements as assigned by the **review leader**. **Check sheets** are completed for each target table. **Actions**, **issues** and **related issues** are documented.

5. The **reviewers** examine the Unmapped Sources Report to verify that all unmapped source columns were intended to remain unmapped.

6. The **reviewers** examine the Unmapped Targets Report to verify that all unmapped target columns were intended to remain unmapped.

7. For **each target table**,
the reviewers compare the **PL/SQL transcripts** to the contents of the **DSIR** to determine if the **number of passes** is correct.
The reviewers enter findings onto the Target Table/Columns Check Sheet.

8. For **each pass**
in each target table, the reviewers compare the PL/SQL transcripts to the contents of the DSIR and examine the sample target data to determine if

the **filter** is correct
the **join condition** is correct
the **row ratio** is correct.

The reviewers enter findings onto the Target Table/Columns Check Sheet.

9. For **each column**
in each pass
in each target table,
the reviewers compare the PL/SQL transcripts to the contents of the DSIR and examine the sample target data to determine if
all **source column dependencies** identified in the DSIR are also listed in the PL/SQL transcripts
the **function,** if any, associated with the column is correct
the **constant column**, if applicable, is properly assigned
the **row specific flag**, if applicable, is properly set.
The reviewers enter findings onto the Target Table/Columns Check Sheet.

10. While filling out the Target Columns Check Sheet for a target table, the **review team** carefully **considers** their actions, issues and related issues to ensure that no actions, if taken, would adversely **impact** any other findings of the review team. **Issues** and **related issues** are discussed to solicit input and possible resolving actions from all review team members.

11. The review team **implements** all actions identified for their respective target tables and columns.

12. The **review leader** reviews all compiled issues and related issues. Data transition requirements **issues** are prioritized for discussion and assignment in the debriefing session. **Related issues** are prioritized and passed on to the **project manager** for action.

Debrief Results from Data Transition Requirements Review

Objectives

1. To **evaluate** the **results** of the review.

2. To identify **corrective actions** that may be required.

3. To communicate **reliable information**
about the **quality** of the data transition requirements to
software project **management**
the **producers** of the data transition requirements and
other affected members of the software engineering group.

Activities

The **review leader**, with assistance from the **recorder**, performs the following activities:

1. Produce the **Technical Review Summary Report**.

2. Produce the **Issues List**.

3. Produce the **Related Issues List**.

4. Deliver the Technical Review Summary Report to
   software project **management**.

5. Deliver the Issues List to
   the **producers** of the data transition requirements.

6. Deliver the Related Issues List to
   software project **management** and
   other affected members of the software engineering group.

APPENDIX G

DATABASE DESIGN PROCEDURES AND CHECKSHEETS

Database Design Review

Purpose

1. To **detect** and **remove defects** from the **database design** specifications **early** and **efficiently**.

2. To develop a better **understanding** of the **database design** specifications and **defects** that might be **prevented**.

3. To achieve more **predictable quality** in the **database design** specifications than can be achieved without a review.

4. To provide project **management** with **reliable information** indicating **progress** in the software development project by **assessing** the database design for **fitness for use** in **subsequent** development activities.

Activities

1. **Prepare** for Database Design Review.

2. **Conduct** Database Design Review.

3. **Debrief Results** from Database Design Review.

Prepare for Database Design Review

Objective

To enable review **participants** to **conduct** a database design **review** with maximum **effectiveness.**

Activities

1. The **producers** of the database objects that will undergo review **verify** that the objects are **ready** for review.
   The producers ensure that all **mandatory properties** of database objects are defined.
   For each **static** table, the producers print a list of **all instances** of data.
   For each **non-static** table, the producers print a list of instances of **at least 10 rows** of data

2. The software project manager selects **participants** for the review to provide appropriate **coverage** of the database objects that will undergo review.

3. The software project manager assigns **roles** (leader, recorder, reviewers) to the participants.

4. The review leader **queries** the participants to determine their **availability** for the review.

5. The review leader reserves a **location** for the review and reserves all necessary **resources** for the review.

6. The review leader **gathers** all necessary review **materials**.

7. The review leader posts a **notice** to the participants that specifies the **time** and **place** for the review and identifies the **database objects** that will undergo review.

8. The review leader **distributes** an agenda and pertinent review **materials** to the participants at least 3 to 5 days before the review.

9. The review leader **queries** all participants at least one day before the review to ensure that each participant remains **available** for the review and each participant is adequately **prepared** for the review.

Prepare for Database Design Review

Check Sheet

Responsible Role	Check Item	✓	Date
Producers	All mandatory properties defined for database objects		
"	List of all instances ready for each static table		
"	List of 10 instances ready for each non-static table		
Project Manager	Review participants identified		
"	Roles assigned to participants		
Review Leader	All participants available for review		
"	Location reserved		
"	All necessary resources reserved		
"	Review materials gathered		
"	Agenda prepared		
"	Participants notified		
"	Agenda and review materials distributed		
	Re-check Item (at least one day prior)	✓	**Date**
Review Leader	Each participant still available for review		
"	Each participant adequately prepared for review		
"	Location still reserved		
"	All necessary resources in place		

Prepare for Database Design Review Check Sheet

Review Materials

Check Item	✓	Date
Comprehensive Entity-Relationship Diagram		
CASE Analysis Object Standards		
CASE Design Data Object Standards		
List of approved abbreviations for CASE objects		
List of Tables to be reviewed		
Table Definition Report from CASE		
Design Phase Check Sheets extracted from CASE*Dictionary		
Mandatory Properties Reports		
List of Values Report		
CDI Constraints Report		
Change Review Board Procedures Document		
For each static table, a list of all instances of data		
For each non-static table, a list of instances of at least 10 rows of data		
Known Issues List		

Prepare for Database Design Review Check Sheet

Resources

Check Item	✓	Date
Meeting room with sufficient tables and chairs		
White board, board markers and eraser		
Easel, flip chart pads, pad markers and masking tape		
Post-it pads		
Note paper, pens, pencils		
File folders for completed check sheets, issue/action forms, summary reports, etc.		
Terminal with access to CASE*Dictionary		
Copies of table analysis check sheets		
Copies of column analysis check sheets		
Copies of issue/action forms		
Copies of issue summary forms		

Prepare for Database Design Review

Mandatory Properties

Mandatory Properties For Tables	✓
Name (length of 24 characters or less)	
DataBase Type (i.e., O for Oracle)	
Alias	
Screen Prompt	
Description	
Primary Key	

Mandatory Properties For Columns	✓
Sequence	
Name (length of 24 characters or less)	
Data Type	
Maximum Length	
Nulls Allowed	
Uppercase Flag	
Display Formatting Information	
Prompt	
Hint Text	
Description	
Mandatory Element Relationships	✓
Either a Domain or Valid Values, or neither, but not both	
For specific domains, Valid Values and Meanings enumerated	
For columns named SEQUENCE NUMBER, a Sequence	

Conduct
Database Design Review

Objectives

1. To **ensure** all **elements** of the database design specifications have been **documented**.

2. To **examine** all **elements** of the database design specifications for **completeness**.

3. To **examine** all **documented elements** of the database design specifications for **conformance to standards.**

4. To **examine** all **elements** of the database design specifications for **conformance to the rules of normalization.**

5. To **correct** identified **defects** in the database design specifications or **document** as **issues** those defects that cannot be quickly resolved.

6. To **document** all **related issues** for the project as a whole not related to the database design specification.

Activities

1. The **review leader** initiates the review by identifying each participant by **role** and ensures that all planned participants are present. All participants affirm their readiness to proceed

2. The **review leader** briefs the group on the **review objectives sequence** of items scheduled start and completion times **logistics** information

3. The **review leader** and a **reviewer** initiate the review by walking through all steps for the **first static table** in front of the group. The review leader ensures that all reviewers understand

   the **conceptual basis** for each **check point**
   the **context** and **criteria** for passing or failing the check point
   how to record **issues** and **related issues**
   Ad-hoc training is conducted as required to ensure that all reviewers are ready to proceed.

4. The **reviewers**, working in teams of two, review all static tables as assigned by the **review leader**. **Check sheets** are completed for each table. **Actions**, **issues** and **related** issues are documented.

5. Upon **completion** of all static table check sheets, each **review team** presents their actions, issues and related issues to other review members. This activity ensures that no actions, if taken, would adversely impact other review team findings. **Issues** and **related issues** are presented briefly to solicit input and possible resolving actions from other review team members.

6. Each review team implements all actions identified for their respective static tables.

7. The **review leader** and a **reviewer** reinitiate the review by walking through all steps for the **first remaining table** in front of the group. The review leader ensures that all reviewers understand

   the **conceptual basis** for each **check point**
   the **context** and **criteria** for passing or failing the check point
   how to record **issues** and **related issues**
   Ad-hoc training is conducted as required to ensure that all reviewers are ready to proceed.

8. The **reviewers**, working in teams of two, review all remaining tables as assigned by the **review leader**. **Check sheets** are completed for each table. **Actions**, **issues** and **related** issues are documented.

9. Upon **completion** of all remaining table check sheets, each **review team** presents their actions, issues and related issues to other review members. This activity ensures that no actions, if taken, would adversely impact other review team findings. **Issues** and **related issues** are presented briefly to solicit input and possible resolving actions from other review team members.

10. Each review team implements all actions identified for their respective tables.

11. The **review leader** reviews all compiled issues and related issues. Database design **issues** are prioritized for discussion and assignment in the debriefing session. **Related issues** are prioritized and passed on to the **project manager** for action.

Conduct
Database Design Review

Check Sheet

Activity	Responsible Role	Check Item	✓
1	Review Leader	All participants are present	
		All participants are prepared	
2	Review Leader	Objectives reviewed by all	
		Review sequence understood by all	
		Schedule understood by all	
		Logistics reviewed	
3	Review Leader	Conceptual basis for static table checkpoints understood by all	
		Context and criteria for each checkpoint understood by all	
		Issue recordation procedures understood by all	
4	Reviewers	All static table check sheets completed	
		All actions, issues, and related issues recorded	
5	Reviewers	All actions are approved or converted to issues	
		All issues are presented to group	
6	Reviewers	All teams complete identified actions	
7	Review Leader	Conceptual basis for remaining table checkpoints understood by all	
		Context and criteria for each checkpoint understood by all	
		Issue recordation procedures understood by all	
8	Reviewers	All remaining table check sheets completed	
		All actions, issues, and related issues recorded	
9	Reviewers	All actions are approved or converted to issues	
		All issues are presented to group	
10	Reviewers	All teams complete identified actions	
11	Review Leader	All issues are reviewed and prioritized for discussion.	
		All related issues reviewed and prioritized.	

Conduct
Database Design Review
Check Sheet

Table/Key Checks

Table Name:

Table Category (circle one or more): Static, Life Cycle, Ord/Seq PK, Other

Table Category	Check Item	✓	Issue/ Action
All	Name, Description, Display Title and Primary Key for the Table are in harmony		
"	The minimum information for creating a row occurrence is meaningful		
"	Table column sequence is consistent with standards		
"	Primary Key is truly unique		
"	Primary Key contains only those components required for uniqueness		
"	All components of the Primary Key are mandatory		
"	Primary Key conveys meaning, and could not be replaced by a Secondary Unique Key that has more value to the customer		
"	Secondary Unique Key, if any, is truly unique		
"	Secondary Key contains only those components required for uniqueness		
Static	Values are static, with a % change of less than 10% annually		
"	Valid business source for values is identified in the table description		
"	Values for a Description column are truly specific and unique		
"	Values for a Description column do not represent the combination of two or more other values for descriptions (i.e., no overlapping meanings)		
"	No other columns exist for the codes and descriptions		
"	Values for a Description column are too long to be considered as a column domain instead		
"	Each value can apply to any target table row related to the static table, and all exceptions are documented		
Life Cycle	Optionality of every column is correctly defined for each state of the row's life cycle		
Ordinal or sequential Primary Key	Table has a separate, logical Unique Key (even if unspecified) that conveys the essence of a row occurrence absent the Primary Key		
"	The Primary Key changes in harmony with the logical Unique Key (i.e., while scanning through row occurrences, when the logical key changes, the primary key changes in harmony)		

Conduct Database Design Review Check Sheet

General Column Check Sheet

Table Name:

Column Name:

Check Item		
Normalization Checks	✓	**Issue/ Action**
1st Normal Form - Column does not repeat or form part of a repeating group.		
2nd Normal Form - If table has a complex primary key, attribute is fully dependent on the entire primary key, not just a component.		
3rd Normal Form - Non-key Column is not completely determined by any other non-key column.		
Generic Domain Checks	✓	**Issue/ Action**
Range - If column values are constrained at an upper or lower limit by business rules, limits are documented.		
Mandatory on Condition - For optional columns, if conditions exist when this column must have a value, these conditions are documented.		
Value Dependency - If column values are constrained by values in other columns, conditions are fully documented. Examples include range dependency, null value dependency, mandatory dependency, complex dependency.		
Finite / Specific Domain Checks	✓	**Issue/ Action**
Each value is not longer than the length of the column		
Each value is clear in meaning, having some mnemonic characteristics		
Descriptions are truly specific and unique		
Descriptions do not represent the combination of two or more other descriptions		
No other columns exist for the codes and descriptions		
Descriptions are not too long to be considered as a domain		
Each value can apply to any related target table row		

Debrief Results from Database Design Review

Objectives

1. To **evaluate** the **results** of the review.

2. To identify **corrective actions** that may be required.

3. To communicate **reliable information**
   about the **quality** of the database design to
   software project **management**
   the **producers** of the database design and
   other affected members of the software engineering group.

Activities

The **review leader**, with assistance from the **recorder**, performs the following activities:

1. Produce the **Technical Review Summary Report**.

2. Produce the **Issues List**.

3. Produce the **Related Issues List**.

4. Deliver the Technical Review Summary Report to
   software project **management**.

5. Deliver the Issues List to
   the **producers** of the database design.

6. Deliver the Related Issues List to
   software project **management** and
   other affected members of the software engineering group.

Debrief Results from Database Design Review

Check Sheet

Responsible Role	Check Item	✓	Date
Review Leader and Recorder	Technical Review Summary Report completed		
"	Issues List completed		
"	Related Issues List completed		
"	All three reports distributed to review participants		
Review participants	Feedback on reports received from all participants		
Review Leader and Recorder	Corrections, if any, incorporated into reports		
"	Technical Review Summary Report delivered		
"	Issues List delivered		
"	Related Issues List delivered		
Affected members of the software engineering group	Official response received by review leader from recipients of Related Issues List		
Review Leader	All reports made available in the visible record of the software development project		

CP Database Design Review

Issue & Action Form

Please be specific
For columns, include Table Name
For UK and FK constraints, include Column Name(s)

Object Type: Table, Column, Constraint (PK, UK, FK), Domain, Other

Object Name:

Related Object Name(s) **Constraint Column Name(s)**

Issue:

Action:

Recorded by: **Date:**

APPENDIX H

WORKSHOP PROCEDURES AND CHECKSHEETS

Module Design Workshop Summary Report

WORKSHOP ID Baseline Management DATE 08 November 1996

STARTING TIME _____

ENDING TIME _____

PARTICIPANTS:

	NAME	SIGNATURE
FACILITATOR:	Sharon Colston	_____
SCRIBE:	Catherine Dougherty	_____
PRESENTER:	Tara Carlson	_____

REVIEWERS:

Sherri Barker	_____	Barbara Newton	_____
Richard Beal	_____	Nancy Patteson	_____
Wendi Dennison	_____	Elaine Powers	_____
Kitty Garza	_____	Mary Snyder	_____
Jeanne Little	_____	Cathy Wood	_____

APPRAISAL OF THE SUBSET DESIGN:

_____ ACCEPTED _____ NO FURTHER WORKSHOP NEEDED

_____ NOT ACCEPTED _____ ANOTHER WORKSHOP REQUIRED

_____ WORKSHOP NOT COMPLETED Reason

_____

SUPPLEMENTARY MATERIALS PRODUCED: DESCRIPTION AND/OR IDENTIFICATION

_____ASSESSMENT CRITERIA CHECKLIST _____

_____DICUSSION ITEM DISPOSITION FORM _____

_____NEXT ACTION _____

_____OTHER _____

Transfer Baseline Processing

BACKGROUND

Q: When can baselines be transferred?

A: Within the development lifecycle of the Baseline only.

The baseline renumbering is automatically done (via foreign keys).

ISSUE TO BE ADDRESSED

However, baseline elements could have been created for any baseline at any time, i.e., not restricted to the CURRENT baseline.

The following is the list of options for transferring baseline elements and further analysis would depend on business rules:

1. Overwrite : "assumes" the official business position is starting from the end of the previous cycle including Source Control Dates, etc.

2. Merge: Requires processing to compare what is in element baselines for both types of baselines. May not be the most cost effective solution since there are only about 7 records on the average per baseline. Further analysis would need to be done

3. Restrict Entry: only allow entry for CURRENT baselines thus eliminating the Merge and Overwrite options.

4. Add to: Automatically add entered information to original contents of baseline elements. The user should be able to decide which records to keep, update, etc.

Example: Transferring Elements from 1.X NOTL to 1.4 D/DL

OTHER RELATED ISSUES THAT MAY IMPACT FINAL DECISION

Is the information captured in the Baseline elements the "OFFICIAL" date information used for scheduling?

Module Design Workshop

Assessment Criteria Checklist

Date scheduled 11/08/1996

Workshop identification Baseline Management

Set the Stage Start time _____

True	False	Check #	Time completed
___	___	1. Agenda presented	_____
___	___	2. Scope and viewpoint for workshop presented	_____
___	___	3. Workshop guidelines presented	_____
___	___	4. Workshop procedure presented	_____
___	___	5. Incoming discussion items listed	_____

Present Business Process Model Start time _____

True	False		Time completed
___	___	6. BPM reflects viewpoint intended for workshop	
___	___	7. BPM suitable context for Module Subset walkthrough	_____

Walkthrough Module Subset (SCRIBE SEE PAGE 2)

Post-Walkthrough Checks Start time _____

True	False		Time completed
___	___	8. All incoming discussion items addressed	_____
___	___	9. Discussion Item Disposition Form completed	_____
___	___	10. Design of Module Subset accepted	_____
___	___	11. Another workshop required	_____
___	___	12. Next Action identified	_____
___	___	13. Design Workshop Summary Report completed	_____
___	___	14. Assessment Criteria Checklist completed	_____

Walkthrough Module Subset Start time _____

New Discussion Items √

True	False		Time completed
___	___	15. Enter Baseline	_____
___	___	16. Enter Baseline Elements	_____
___	___	17. Enter Baseline Lineage	_____
___	___	18. Transfer Baseline Relationships	_____
___	___	19. Print CP Baselines Listing Report	_____
___	___	20. Print Baseline Count and List of Open CPCRs Report	_____
___	___	21. Print Baseline Lineage Path Report	_____
___	___	22. Print Baseline Definition Report	_____

Assessment Criteria Checklist

Supplemental Notes

Record the number from the assessment criteria checklist along with clarifying notes.

Check #	Notes
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

Module Design Workshop Agenda

Workshop: Baseline Management Workshop

Date: November 8, 1996
Vitro Media Room

Attendees:

Sherri Barker	_____	Barbara Newton	_____
Richard Beal	_____	Nancy Patteson	_____
Wendi Dennison	_____	Elaine Powers	_____
Kitty Garza	_____	Mary Snyder	_____
Jeanne Little	_____	Cathy Wood	_____

1. Workshop Introduction and Welcome — S. Colston
2. Workshop Procedures — S. Colston
3. Acknowledgement of Incoming Discussion Items — S. Colston
4. Baseline Overview Briefing — T. Carlson
5. Presentation of Business Process Model — T. Carlson
6. Identification of Module Subset — T. Carlson
7. Module Subset Walkthrough — T. Carlson
   - ☐ **Enter Baseline**
   - ☐ **Enter Baseline Elements**
   - ☐ **Enter Baseline Lineage**
   - ☐ **Transfer Baseline Relationships**
   - ☐ **Print CP Baselines Listing Report**
   - ☐ **Print Baseline Count and List of Open CPCRs Report**
   - ☐ **Print Baseline Lineage Path Report**
   - ☐ **Print Baseline Definition Report**
8. Post Workshop Administration — C. Dougherty

Module Design Post-Workshop

Closeout Checklist

Date completed _____

Workshop identification_____

True	False	Check #	Date completed
___	___	1. Approval signatures obtained from participants	_____
___	___	2. Incoming Issues Closed	_____
___	___	3. Issues entered in tracking system	_____
___	___	4. Process Models Updated	_____
___	___	5. Process Models Peer Review QA	_____
___	___	6. Disposition of discussion items prepared	_____
___	___	7. Discussion item disposition QA and ready to distribute	_____
___	___	8. Participants briefed on ***OUTSTANDING ISSUES***	_____
___	___	9. Statistics prepared	_____
___	___	10. Statistics peer reviewed QA and ready to distribute	_____
___	___	11. Functional requirements specification prepared	_____
		12. Functional requirements specification ready to distribute	_____
___	___	13. Post-workshop materials distributed	_____
___	___	14. Module subset software updated (CASE included!)	_____
___	___	15. Screen design reflects workshop items	_____
___	___	16. Module subset signature testing completed	_____
___	___	17. Module subset peer reviewed	_____
___	___	18. Module subset available for customer liaison beta testing	_____
___	___	19. Module specifications prepared	_____

APPENDIX I

TEST PROCEDURES

ACCESS BL/2 USER ACCEPTANCE TEST

Name:_____ **Username:**_____ **Test Date:**_____

Signature: _____ **Test Role:**_____ **Test Case:**_____

Instructions: These procedures are intentionally meant to be general and you should expect to have certain steps which cannot be completed. Try to complete each step. All steps marked "NO" require a remark. Record all remarks in either the space provided, or use the additional remarks sheets. Please use the corresponding Test key to verify your results.

STEP NUMBER	IMPLEMENT CHANGE REQUEST	STEP COMPLETED?	
		YES	NO
1.0	Select [SC] from the Forms Menu		
	REMARKS:		
2.0	Insert new change request.		
	REMARKS:		
3.0	Verify Change Request entered/modified is correct		
	REMARKS:		
3.1	Query Change Request entered		
	REMARKS:		
3.2	Run report		
	REMARKS:		
4.0	Release the Change Request to the ECCB		
	REMARKS:		
5.0	Schedule an ECCB board meeting		
	REMARKS:		
6.0	Schedule the Change Request for the ECCB Meeting		
	REMARKS:		
7.0	Package the Change Request with other Changes for Boarding		

ACCESS BL/2 USER ACCEPTANCE TEST

	REMARKS:		
7.1	Create Change Package		
	REMARKS:		
7.2	Assign Change Request to Change Package		
	REMARKS:		
7.3	Schedule Change Package for ECCB		
	REMARKS:		
7.4	Unschedule the standalone change request		
	REMARKS:		
8.0	Verify the change request is scheduled for ECCB		
	REMARKS:		
8.1	Query scheduled change request		
	REMARKS:		
8.2	Print Agenda Summary Report		
	REMARKS:		
9.0	Review the change request [do not release to ECCB or mark as final yet, may have multiple reviews]		
	REMARKS:		
10.0	Verify Change Review entered		
	REMARKS:		

ACCESS BL/2 USER ACCEPTANCE TEST

10.1	Query Change Review		
	REMARKS:		
10.2	Run report		
	REMARKS:		
11.0	Release Change Review to ECCB		
	REMARKS:		
12.0	Verify Change Review released to ECCB		
	REMARKS:		
12.1	Query Change Review		
	REMARKS:		
12.2	Run ECCB Agenda Report		
	REMARKS:		
13.0	Status the Change Request as a result of the ECCB meeting		
	REMARKS:		
13.1	Add CPC with Board Action		
	REMARKS:		
13.2	Create Board Action for an existing CPC		

ACCESS BL/2 USER ACCEPTANCE TEST

	REMARKS:		
14.0	Verify Board Actions entered/modified is correct [hint: query and run a report to verify]		
	REMARKS:		
14.1	Query Board Action [view CPCs]		
	REMARKS:		
14.2	Run Board Action Status report		
	REMARKS:		
15.0	Assign Action Item(s) from the ECCB [investigate the Change Review, do a new review, coordinate with another element, etc.]		
	REMARKS:		
16.0	Verify Action Items entered/modified are correct		
	REMARKS:		
16.1	Query Action items		
	REMARKS:		
16.2	Run report		
	REMARKS:		
17.0	Designate a final Change Review		
	REMARKS:		
17.1	Query Change Review		
	REMARKS:		
17.2	Update and mark as final		

ACCESS BL/2 USER ACCEPTANCE TEST

	REMARKS:		
18.0	Release the Change Request to the CRB		
	REMARKS:		
19.0	Run ECCB Minutes/Recommendation Package Report		
	REMARKS:		
20.0	Update the Change Request and assign a CRB Identification Number		
	REMARKS:		
21.0	Schedule a CRB board meeting		
	REMARKS:		
22.0	Schedule Change Request/package for CRB meeting		
	REMARKS:		
23.0	Prepare CRB Agenda		
	REMARKS:		
23.1	Run CRB Minutes Report		
	REMARKS:		
23.2	Run CRB Action Items Report		
	REMARKS:		
24.0	Status the Change Request(s) as a result of the CRB meeting		
	REMARKS:		
24.1	Add CPC with Board Action		

ACCESS BL/2 USER ACCEPTANCE TEST

	REMARKS:		
24.2	Create Board Action for an existing CPC		
	REMARKS:		
25.0	Verify Board Actions entered/modified is correct [hint: query and run a report to verify]		
	REMARKS:		
25.1	Query Board Action [view CPCs]		
	REMARKS:		
25.2	Run Board Action Status report		
	REMARKS:		
26.0	Assign Action Item(s) from the CRB [investigate the Change Review, do a new review, coordinate with another element, etc.]		
	REMARKS:		
27.0	Verify Action Items entered/modified are correct		
	REMARKS:		
27.1	Query Action items		
	REMARKS:		
27.2	Run report		
	REMARKS:		
28.0	Run CRB Minutes Report		
	REMARKS:		
29.0	Transfer Change Request to another Element		

ACCESS BL/2 USER ACCEPTANCE TEST

	REMARKS:		
30.0	Update Change Request information		
	REMARKS:		
31.0	Delete Change Request		
	REMARKS:		
32.0	Update Change Review		
	REMARKS:		
33.0	Delete Change Review		
	REMARKS:		
34.0	Query any Change Request		
	REMARKS:		
35.0	Query any Change Review		
	REMARKS:		

ACCESS BL/2 USER ACCEPTANCE TEST

Instructions: Please indicate the Step number and enter any additional comments in the remarks section.

Step Number:	
Remarks:	
Step Number:	
Remarks:	

ACCESS BL/2 USER ACCEPTANCE TEST

CRB CM TEST KEY

Step Number	Expected Response	
1.0	Y	
2.0	Y	
3.0		N
4.0	Y	
5.0	Y	
6.0	Y	
7.0	Y	
8.0	Y	
9.0	Y	
10.0	Y	
11.0		
12.0		
13.0		
14.0		
15.0		
16.0		
17.0		
18.0		
19.0		
20.0		
21.0		
22.0		
23.0		
24.0		
25.0		
26.0		
27.0		
28.0		
29.0		
30.0		
31.0		
32.0		
33.0	Y	

DISTRIBUTION

	Copies
DOD ACTIVITIES (CONUS)	
ATTN CODE A76 (TECHNICAL LIBRARY) COMMANDING OFFICER CSSDD NSWC 6703 W HIGHWAY 98 PANAMA CITY FL 32407-7001	1
DEFENSE TECH INFORMATION CTR 8725 JOHN J KINGMAN RD SUITE 0944 FORT BELVOIR VA 22060-6218	2
NON-DOD ACTIVITIES (CONUS)	
THE CNA CORPORATION P O BOX 16268 ALEXANDRIA VA 22302-0268	1
INTERNAL	
B60 (TECHNICAL LIBRARY)	3
N	1
N05	1
N055	1
N10	1
N20	1
N20C	1
N21	1
N25	1
N27	1
N27 (COLSTON)	25
N60	1
N80	1
N90	1